НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК_____

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

Магістерська дисертація

зі спеціальності - 122 Комп’ютерні науки
за спеціалізацією – Комп’ютерний моніторинг та геометричне моделювання процесів і систем
на тему: Інструментальні засоби моделювання систем автоматизації споруд

Виконав (-ла): студент (-ка) 6 курсу, групи ТМ-81мп
Олійник Сергій Петрович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.ф,-м.н, Тарнавський.Ю.А _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ - 2019

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 122 Комп’ютерні науки
за спеціалізацією – Комп’ютерний моніторинг та геометричне моделювання процесів та систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В._____
(прізвище, ініціали) (підпис)

«____» _____ 2019р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Олійник Сергій Петрович
(прізвище, ім’я, по батькові)

1. Тема дисертації Інструментальні засоби моделювання систем автоматизації споруд_____
_____

Науковий керівник к.ф.-м.н., доцент Тарнавський Ю.А. _____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “04” листопада 2019 року № 3812-с

2. Строк подання студентом дисертації “ ” _____ 2019 року

3.Об’єкт дослідження системи автоматизації споруд_____
_____

4. Предмет дослідження інструменти для моделювання систем автоматизації споруд_____

5. Перелік питань, які потрібно розробити _____

1) проаналізувати сучасні системи автоматизації споруд;
2) розробити модель моніторингу середовища систем автоматизації споруд;
3) розробити модель бібліотеки візуальних компонентів;
4) розробити архітектуру системи;
5) розробити програмне забезпечення.
6. Орієнтований перелік ілюстративного матеріалу
мета, постановка задачі, структура системи, структура клієнтської та серверної частин, структура та модель обробки повідомлень
7. Орієнтований перелік публікацій

8. Дата видачі завдання "__" _____20___р .

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Затвердження теми роботи	30.11.2018	
2	Вибір напрямку наукового дослідження	12.12.2018	
3	Аналітичний огляд літератури	15.02.2019	
4	Збір та аналіз теоретичного матеріалу по темі дослідження	05.03.2019	
5	Розробка архітектури та загальної структури системи	06.09.2019	
6	Розробка структур окремих підсистем	15.10.2019	
7	Програмна реалізація системи	27.10.2019	
8	Оформлення пояснювальної записки	15.11.2019	
9	Захист програмного продукту	23.10.2019	
10	Попередній захист магістерської дисертації	20.11.2019	
11	Захист магістерської дисертації		

Студент _____ Олійник С.П.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Тарнавський Ю.А.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

на магістерську дисертацію

Структура й обсяг дипломної роботи. Магістерська дисертація складається зі вступу, 6 розділів, висновку, переліку посилань з 22 найменування, 1 додатку, і містить 30 рисунків, 22 таблиці. Повний обсяг магістерської дисертації складає 87 сторінок, з яких перелік посилань займає 2 сторінки, додатки –5 сторінок.

Актуальність теми. Кожна людина прагне покращити умови проживання у власному житлі, роблячи його безпечнішим та більш комфортним. За допомогою новітніх технологій, люди можуть собі дозволити управляти своїм житлом через програмний додаток, тим самим заощаджуючи свій час та роблячи умови проживання більш. Розробка систем автоматизації споруд є клопітким процесом і потребує інструментів для тестування, проведення досліджень та експериментів. Розробка інструментальних засобів моделювання систем автоматизації споруд допоможе полегшити процес розробки даних систем.

Мета дослідження полягає в створенні інструментальних засобів моделювання систем автоматизації споруд з метою дослідження їх якості і ефективності.

Завдання дослідження. Для досягнення поставленої задачі були поставлені наступні завдання:

- розробити бібліотеку візуальних компонентів для систем автоматизації споруд;
- створити засоби для розробки сценарію керування пристроями автоматизованої споруди;
- забезпечити збереження даних моделі та синхронізацію з пристроями;
- виконати тестове моделювання системи автоматизації споруди.

Об’єктом дослідження моделювання систем автоматизації споруд

Предметом дослідження є інструменти для моделювання систем автоматизації споруд

Наукова новизна одержаних результатів. Наукова новизна полягає у створенні нових моделей управління пристроями за допомогою бібліотеки візуальних компонентів.

Практичне значення. Інструменти дозволяють створити тестову модель системи автоматизації споруд та задати режим роботи для кожного пристрою, який міститься в даній моделі.

Ключові слова: *ІНСТРУМЕНТАЛЬНІ ЗАСОБИ, МОДЕЛЮВАННЯ, АВТОМАТИЗАЦІЯ СПОРУД.*

ABSTRACT

for a master's dissertation

Structure and volume of the thesis. The master's dissertation consists of an introduction, 6 sections, conclusion, a list of links of 22 titles, 1 annex, and contains 30 pictures, 22 tables. The full volume of the master's thesis is 87 pages, of which the list of links occupies 4 pages, the applications - 5 pages.

Actuality of theme. Everyone tryes to improve living conditions in their own home, making it safer and more comfortable. With the help of the latest technologies, people can afford to manage their homes through a software application, thus saving their time and making their living conditions more comfortable. The development of building automation systems is a tedious process and requires tools for testing, research and experimentation. Development of tools for modeling of systems automation of structures will help to facilitate the process of development of these systems.

The purpose of the study is to create tools for modeling of building automation systems in order to investigate their quality and efficiency.

Objectives of the study. To achieve this task, the following tasks were set:

- develop a library of visual components for building automation systems;
- to create the means for developing the scenario of control of devices of automated construction;
- ensure that model data is stored and synchronized with devices;
- perform test simulation of the building automation system.

The object of study is the modeling of systems of building automation

The subject of the study are tools for modeling of building automation systems

Scientific novelty of the obtained results. The scientific novelty is to create new models of device management using a library of visual components.

Practical meaning. The tools allow you to create a test model of the building automation system and set the operating mode for each device contained in this model.

Keywords: *TOOLS, MODELING, AUTOMATION OF STRUCTURES*.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ПОЗНАЧЕНЬ

БД – база даних

СКДБ – система керування базами даних

Фреймворк - інфраструктура програмних рішень, що полегшує розробку складних систем

САПР – система автоматизованого керування.

ВСТУП

Кожна людина прагне покращити умови проживання у власному житлі, роблячи його безпечнішим та більш комфортним. За допомогою новітніх технологій, люди можуть собі дозволити управляти своїм житлом через програмний додаток, тим самим заощаджуючи свій час та роблячи умови проживання більш комфортними.

Актуальність теми дипломної роботи полягає в тому, щоб дати змогу користувачеві створити макет з набором функцій для керування та контролю пристроями в певному приміщенні. За допомогою візуальних компонентів, користувач може виставити на плані будинку певні візуальні елементи, які відповідають певним пристроям. Зручний інтерфейс дозволяє легко управляти кожним пристроєм, даючи змогу створювати, видаляти і переміщати графічний елемент, а також змінювати режим його роботи (вмикати та вимикати) або встановити дату та час зміни режиму роботи пристрою. Інформація про кожен елемент зберігається в базі даних, що дає змогу відновити інформацію у разі її видалення чи пошкодження.

Даний програмний продукт дає можливість створювати макет з електричними пристроями для певної будівлі та робити певні досліди, наприклад, приблизно обрахувати споживання електроенергії за певний проміжок часу. На основі обчислень користувач може отримати декілька звітів та зробити певний порівняльний аналіз. Основні переваги даних систем:

— економія часу;

— повний контроль за будинком;

— економне використання освітлення та опалення;

— збереження даних в базі даних.

Для досягнення мети необхідно виконати наступні завдання:

— розробити бібліотеку візуальних компонентів для систем автоматизації споруд;
— створити засоби для розробки сценарію керування пристроями автоматизованої споруди;
— забезпечити збереження даних моделі та синхронізацію з пристроями;
— виконати тестове моделювання системи автоматизації споруди.

Для реалізації клієнтської частини даного програмного забезпечення було обрано мову програмування C++ на основі фреймворка Qt, що дає змогу розробляти графічний додаток за швидкий період часу та для різних платформ. Для розробки серверної частини було обрано мову програмування Python. Для керування базою даних була вибрана документо-орієнтована система керування базами даних MongoDB.

У першому розділі пояснювальної записки сформульовано мету й описано постановку задачі. Також представлено основні пункти для реалізації даного програмного продукту та описано функціонал, який повинна включати програма.

У другому — розділі наводяться приклади схожих реалізованих програмних продуктів. Описана характеристика та наведено переваги і недоліки кожної програми. В даному пунті приведено приклади основних систем, з якими може зіткнутись користувач при розробці інструментальних засобів.

У третьому розділі описано архітектуру даної програми. Приведена характеристика всіх використаних шаблонів проектування з перевагами та недоліками та обґрунтовано причину їх вибору.

У четвертому розділі описується програмна реалізація створення програми. Детально описано всі технології, які використовувались для створення даного програмного продукту.

У п’ятому розділі подано методику роботи користувача з програмною

системою. Наведено приклади використання всіх можливих візуальних компонентів.

У шостому розділі описано економічне обґрунтування реалізації стартап-проекту на дану тему.

1. ЗАДАЧА СТВОРЕННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ МОДЕЛЮВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ СПОРУД

Метою розробки є створення інструментальних засобів моделювання систем автоматизації споруд

Для досягнення поставленої мети необхідно:

— розробити бібліотеку візуальних компонентів для систем автоматизації споруд;

— створити засоби для розробки сценарію керування пристроями автоматизованої споруди;

— забезпечити збереження даних моделі та синхронізацію з пристроями;

— виконати тестове моделювання системи автоматизації споруди.

Програмний додаток має включати наступний функціонал:

— можливість створювати візуальні компоненти із доступного каталогу;

— можливість видаляти візуальний компонент;

— можливість переміщувати візуальний компонент;

— візуалізувати робочий статус кожного компонента в системі;

— можливість змінювати режим роботи пристрою за допомогою візуального компонента;

— можливість задати дату та час зміни режиму роботи пристрою;

— відсилати повідомлення про зміну режиму роботи певного пристрою на сервер;

— зберігати інформацію про кожен компонент системи локально в json файлі;

— .зберігати дані про кожен компонент в базі даних.

Кожному пристрою в даній системі відповідає компонент з унікальною адресою. Даний компонент дозволяє користувачу вмикати чи вимикати відповідний пристрій як миттєво так і через певний проміжок часу. Інформація про кожен компонент зберігається на стороні клієнта і дублюється в базі даних на стороні сервера.

За розміщенням компонента на плані будівлі можна легко дізнатись реальне місце знахожнення приладу, якому відповідає даний компоннет. Комунікація між клієнтом та сервером відбувається через інтернет за допомогою протоколу TCP/IP.

Основними користувачами програмної системи являються:

— особи, котрі прагнуть використовувати дану систему в себе вдома;

— підприємці, котрі займаються розробкою систем автоматизації споруд;

— особи, котрі прагнуть поставити певний експеримент.

2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

На даний момент існує багато систем розумного дому, які автоматизовують певні процеси у приміщенні. Серед них є як одиничні системи, які складаються з пристроїв одного типу, так і комплексні, які вміщують в собі пристрої різних типів. Далі пропонується опис програмних рішень для автоматизації приміщень.

Система розумного дому дає змогу легко керувати пристроями та постійно регулює різні події в будинку. За рахунок оптимального використання ресурсів, користувач може заощадити на опаленні та електроенергії в будинку. Дана система дозволяє легко налаштувати роботу пристроїв під розпорядок дня користувача.

2.1 Програмна система Smart Things

Smart Things — це програма, розроблена компанією Samsung, для централізованого курування будинком. Дана програма спроможна миттєво підключаються до різних датчиків, замків, вимикачів та інших сумісних пристроїв. Програма спроможна співпрацювати з пристроями від різних виробників.

Дана система спроможна:

— співпрацювати з пристроями, та синхронізувати їх роботу;

— контролювати якість освітлення, температуру тощо;

— запобігати взлому;

— керувати пристроями на відстані.

Основними недоліками є:

— не всі пристрої співпрацюють з даною системою;

— працює лише за наявності інтернет зв’язку.

Інтерфейс системи зручний і дозволяє легко управляти як одним так і групою пристроїв одночасно. Комунікація між пристроями і користувачем відбувається через Wi-Fi з’єднання (рисунок 2.1).

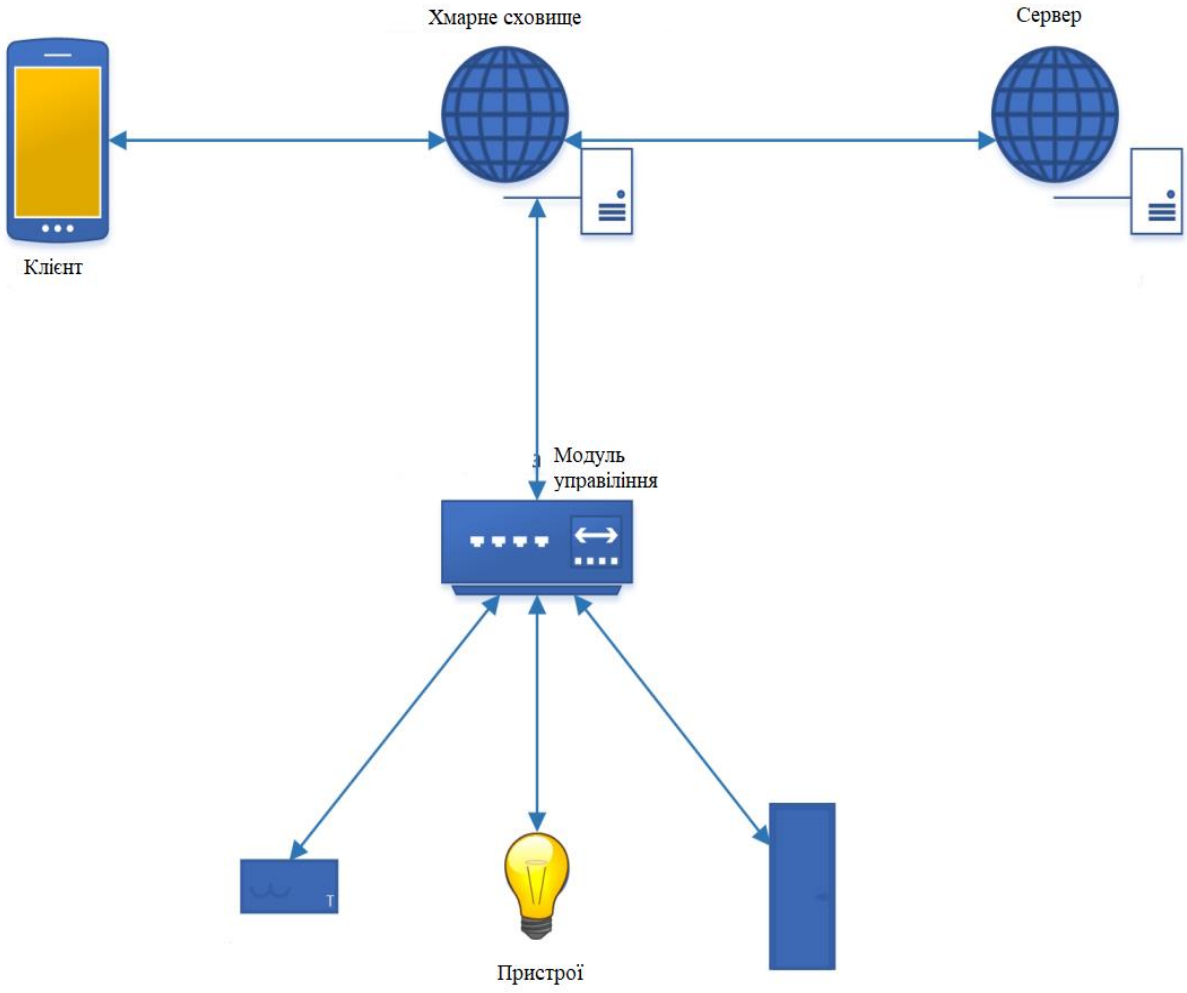


Рисунок 2.1 — Комунікація між пристроями та користувачем в системі Smart Things

В даній системі користувач управляє пристроями за допомогою програмного додатка. Змінивши параметр роботи певного пристрою, програма

відправляє сигнал до модулю управління пристроями, який приймає сигнал та змінює режим роботи відповідного пристрою. Інформація про користувача зберігається на сервері, а інформація про пристрої в хмарному сховищі.

2.2 Програмна система Apple HomeKit

Apple HomeKit — це програмне забезпечення, розроблене компанією Apple, яке дозволяє користувачам налаштовувати пристрій для управління будинком. Основною перевагою даної системи є надійний захист інтелектуального будинку від вторгнення та несанкціонованого доступу сторонніх осіб.

Програма дозволяє управляти пристроями трьома основними способами: режим «Home», режим «Room» та режим «Automation». Режим «Home» представляє собою список пристроїв, якими користувач найчастіше користується. За допомогою режиму «Room» управляються пристрої, які оточують будинок. Режим «Automation» дозволяє автоматизувати інтелектуальну поведінку пристроїв та систем у будинку, наприклад, увімкнути сигналізацію чи вимкнути світло при виході з приміщення. Користувач може створювати окремі режими, в яких декілька пристроїв будуть працювати як єдина автоматизована система.

Нові пристрої можна додати, скануючи код із шести цифр, проте користувачу слід окремо зберігати дані коди на випадок коли знову прийдеться додавати пристрій в систему. Також клієнт може створювати нові коди для деяких пристроїв, але це не легка процедура, яка потребує сторонніх додатків.

Основними недоліками є:

— дана програма підтримує лише операційну систему IOS;

— складно під’єднати нові пристрої до домашньої системи.

Комунікація між пристроями і користувачами відбувається через Wi-Fi з’єднання (рисунок 2.2).

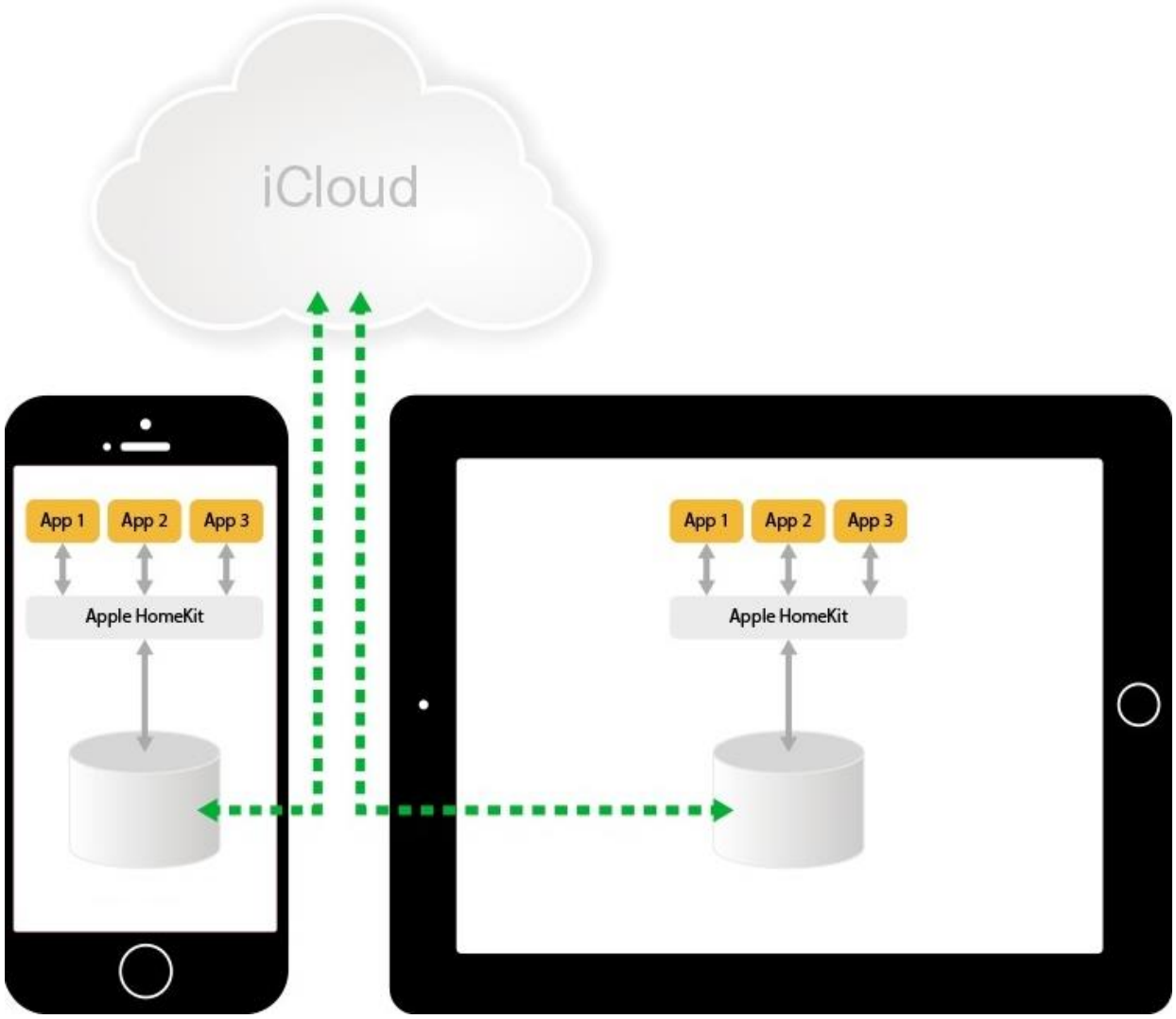


Рисунок 2.2 — Комунікація між пристроями та користувачами в системі Apple HomeKit

Принцип роботи даної системи схожий з попереднім прикладом. Ідея полягає в тому, що вся інформація про пристрої та режими їх роботи зберігається в сховищі. При входженні в систему дані синхронізуються і користувач має змогу бачити інформацію по кожному пристрою.

2.3 Програмна система Nest

Nest — це система, розроблена компанією Nest, яка дозволяє централізовано управляти приладами у будинку. Даний продукт легко співпрацює з іншими платформами (окрім HomeKit), що дає змогу користувачу без залучення додаткового програмного забезпечення, підключитися до різних наборів пов’язаних технологій. Даний продукт дозволяє легко керувати різними побутовими системами в будинку такими як: система освітлення, система вентиляції, опалення, відеоспостереження тощо.

2.4 Допоміжні програмні системи

Програмні продукти зазначені вище являються найбільш популярними системами, з якими користувач може зіткнутись при виборі системи для автоматизації будинку. Але серед них є й інші компанії, які створюють свій власний продукт, розширюючи спектр програмних та апаратних параметрів за потребою користувача. Нижче приведені підсистеми системи, співпрацюють чи являються альтернативою системам, зазначеним вище.

Wink Hub — це модуль управління пристроями, який приймає сигнали від користувача і безпосередньо керує пристроями. Даний концентратор підтримує дуже велику кількість інтелектуальних протоколів від різних виробників.

IFTTT — це веб-служба, яка дозволяє різним платформам і гаджетам взаємодіяти між собою. Багато платформ використовують дану технологію для комунікації між пристроями та для усунення певних технічних проблем. Основними конкурентами даної системи є програмний продукт Strignify, який за допомогою потоків може управляти декількома діями, та Yonomi — домашній концентратор, який вбудований в смартфон (як iOS, так і Android), і

дозволяє керувати розумними домашніми пристроями, синхронізувати їх і працювати разом у тандемі.

В таблиці 2.1 наведено порівняння існуючих програмних засобів з розробленим програмним продуктом.

Таблиця 2.1 — Порівняння існуючих програмних засобів.

Назва продукту	Характеристика
Smart Things	Дає змогу керувати пристроями на відстані та синхронізувати їх роботу. Система залежна від інтернет мережі. Співпрацює лише з пристроями, які підтримують дану платформу.
Apple HomeKit	Дає змогу синхронізувати та керувати пристроями із загального центру керування. Відзначається надійним захистом від сторонніх користувачів. Призначений виключно для пристроїв операційної системи IOS.
Nest	Співпрацює з широким асортиментом пристроїв. Легко підключитися до нового пристрою без підтримки стороннього програмного забезпечення.

3. ПРОЕКТУВАННЯ ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ МОДЕЛЮВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ СПОРУД

Дана програма побудована на основі архітектурного шаблону MVC (Модель – Вид – Контролер). Архітектурний шаблон MVC поділяє програму на три частини (рисунок 3.1) [11]. У тріаді до обов'язків компоненту Модель (Model) входить зберігання даних і забезпечення інтерфейсу до них. Вигляд (View) відповідальний за представлення цих даних користувачеві. Контролер (Controller) керує компонентами, отримує сигнали у вигляді реакції на дії користувача, і повідомляє про зміни роботи компоненту модель. Така внутрішня структура в цілому поділяє систему на самостійні частини і розподіляє відповідальність між різними компонентами.

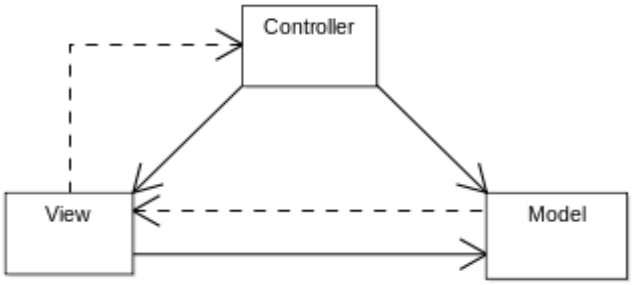


Рисунок 3.1 — Шаблон MVC

Модель, як вже було відмічено, інкапсулює ядро даних і основний функціонал з їх обробки. Також компонент модель не залежить від процесу введення або виведення даних. Компонент виводу може мати декілька взаємопов'язаних областей, наприклад, віджети і поля форм, в яких відображається інформація. У функції контролера входить моніторинг за

подіями, що виникають в результаті дій користувача (зміна положення курсору миші, натиснення кнопки або введення даних в текстове поле).

В даному випадку даний продукт не є цілісною системою, а лише являється контролером. Основна задача контролера передавати запит від клієнта до сервера, який в свою чергу записує дані до бази даних та відправляє сигнал модулю управління пристроями. Модуль управління пристроями являється грає роль моделі і безпосередньо відповідає за керування пристроями та побутовими системами. Блок керування пристроями приймає повідомлення про зміну роботи певного пристрою. Кожне повідомлення містить в собі ідентифікаційний код пристрою, який є ідентичним з кодом візуального компонента та значення візуального компонента. Кожен компонент може передавати інформацію про ввімкнення або вимкнення пристроїв або побутових систем, але не повідомляє користувача про поломку певного пристрою. Передача сигналу між контролером та сервером відбувається через інтернет по протоколу TCP/IP, тому без інтернет зв'язку неможливо успішно синхронізуватись між модулем управління пристроями і користувачем і система являється не робочою. Структура системи відображається на рисунку 3.2.

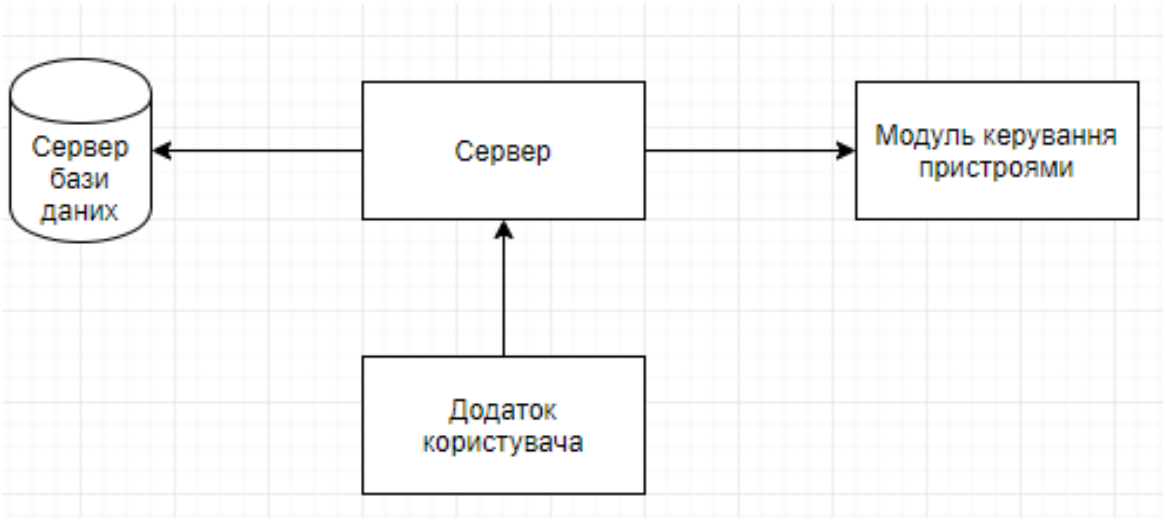


Рисунок 3.2 — Структура системи

Інформація про зміну статусу пристрою відправляється серверу, де повідомлення зчитується і визначається який конкретно пристрій треба включити, виключити чи змінити режим його роботи. Залежно від повідомлення, сервер може відразу відправити сигнал про зміну статусу пристрою, або зробити це в певний період дати та часу.

Користувач відправляє моделі запити для управляння пристроями та отримує результати запитів через контролер. Інтерфейс користувача представлений у вигляді плану будинку. Користувач може додавати нові компоненти, видаляти існуючі, задавати параметри компонентам, міняти їх координати на плані. На рисунку 3.3 представлено діаграму прецедентів користувача [21].

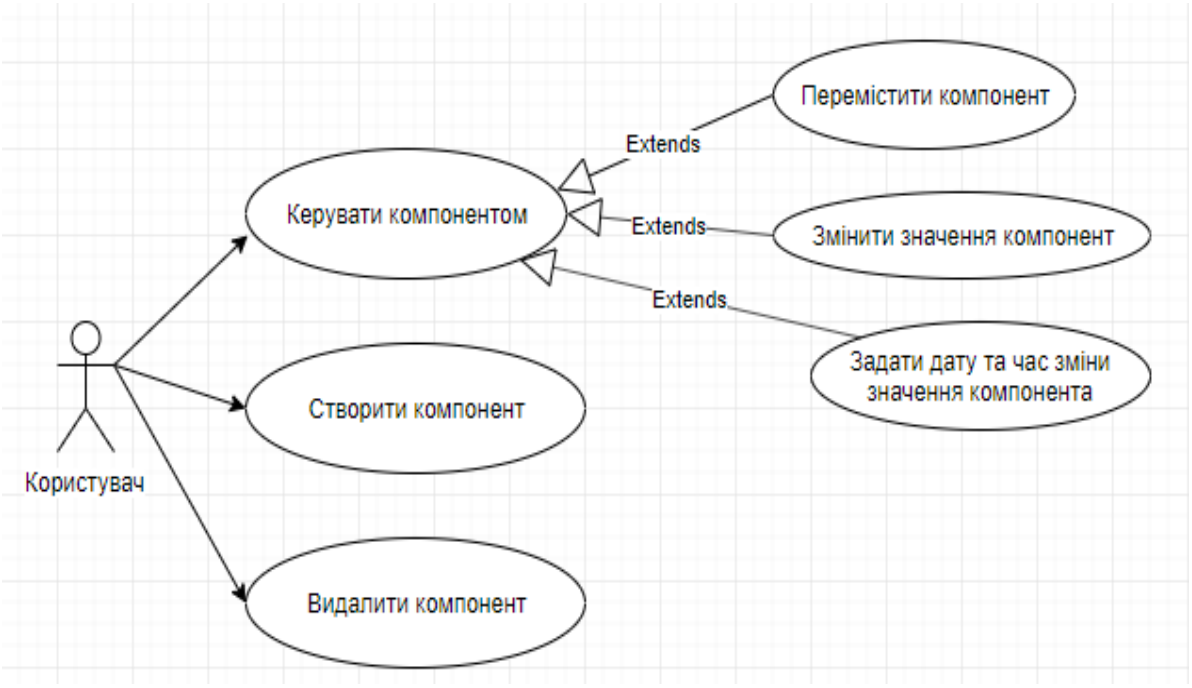


Рисунок 3.3 — Діаграма прецедентів користувача

Клієнт містить реалізацію лише користувацького інтерфейсу та не займається безпосередньою роботою з пристроями, оскільки ці обов’язки бере на себе модуль управління пристроями.

3.1 Проектування бібліотеки візуальних компонентів

Користувач може створювати та керувати компонентами через інтерфейс. Створення повністю нового компонента та модернізація моделі відбувається лише програмним шляхом.

В даній роботі користувач може створювати лише компоненти із представленого йому списку.

До списку входять:

— проста лампа;

— складна лампа;

— вентилятор;

— двері;

— жалюзі;

— нагрівач.

Кожному компоненту з переліченого вище списку створено відповідно клас, який успадковується від батьківського класу Device [1]. Створення візуального компонента відбувається за допомогою породжуючого шаблону Factory Method, де для певного компонента викликається відповідний прототип [4].

Прототип — певний об’єкт, здатний створювати копію самого себе (клонувати). В даному шаблоні абстрактний клас містить набір параметрів та властивостей, які притаманні всім похідним підкласам. В абстрактному класі об’являється метод clone(), який перевизначається в усіх похідних класах. В даному випадку для створення копій можна використовувати конструктор копіювання з реалізацією глибокого копіювання об’єктів [2]. Зазвичай всі існуючі в системі прототипи зберігаються в спеціальній колекції, представленій у вигляді асоціативного масиву. Реєстр прототипів дає змогу добавити чи видалити прототип, а також створити об’єкт по ідентифікатору типу. Динамічне

додавання та видалення прототипів додає гнучкості системі. UML діаграма шаблону Prototype наведена на рисунку 3.4.

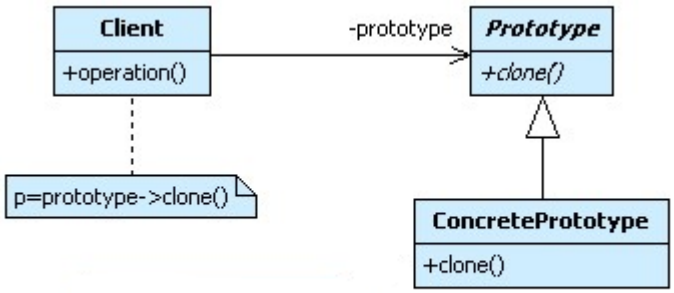


Рисунок 3.4 — UML-діаграма класів шаблону Prototype.

На даній діаграмі в абстрактному класі Prototype проініціалізовано абстрактний метод clone. Клас ConcretePrototype успадковується від базового класу та перевизначає метод clone. В класі Client для створення об’єкту конкретного класу визивається відповідний метод clone, внаслідок чого створюється копія базового класу Prototype та ініціалізуються нові властивості класу ConcretePrototype.

Шаблон Prototype використовується:

— у випадку, коли система повинна оставатися незалежною як від процесу створення об’єкту, так і від типів породжуючих об’єктів;

— необхідно створювати об’єкти, точні класи яких відомі вже на стадії виконання програми.

Основним недоліком даного шаблону є те, що кожен тип створюваного об’єкта повинен реалізовувати операцію клонування і в певних випадках глибоке копіювання являється доволі складною задачею.

Шаблон Factory Method використовується в наступних випадках:

— система повинна бути незалежною від того, як компоненти утворюються, складаються та представляються;

— система може розширятись шляхом додавання об’єктів нових типів;

— заздалегідь відомо, коли потрібно створювати об'єкт, але невідомий його тип.

Існує два варіанти реалізації шаблону Abstract Method [8]:

— узагальнений конструктор, коли в базовому класі, від якого успадковують похідні класи, визначається статичний фабричний метод. Як параметр в цей метод повинен передаватися ідентифікатор типу створюваного об'єкта (рисунок 3.5);

— класичний варіант фабричного методу, коли інтерфейс фабричних методів оголошується в незалежному класі-фабриці, а їх реалізація визначається конкретними підкласами цього класу (рисунок 3.6).

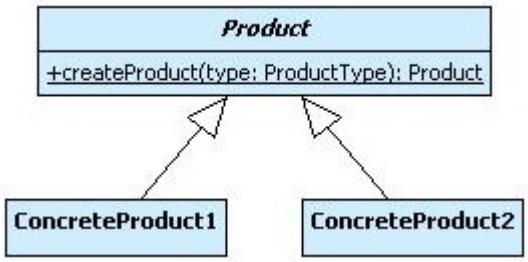


Рисунок 3.5 — UML-діаграма класів шаблону Abstract Method. Узагальнений конструктор

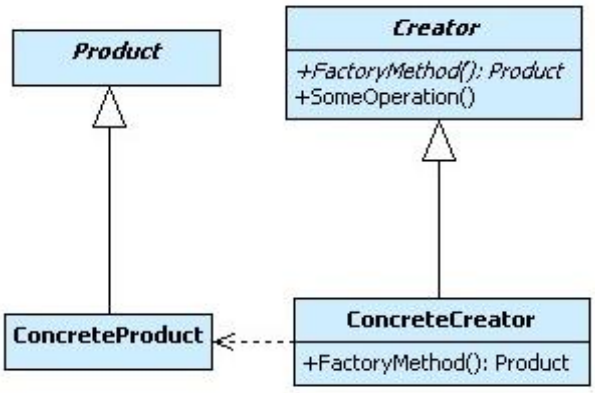


Рисунок 3.6 — UML-діаграма класів шаблону Abstract Method. Узагальнений конструктор

Класичний варіант шаблону Factory Method використовує ідею поліморфної фабрики. Спеціально виділений для створення об'єктів поліморфний базовий клас Creator оголошує інтерфейс фабричного методу factoryMethod, а похідні класи його реалізують.

Основна перевага шаблону Factory Method полягає в тому, що створення об'єктів різних типів залишається незалежною від самого процесу створення і від типів створюваних об'єктів.

Недоліком шаблону є те, що навіть для породження єдиного об'єкта необхідно створювати відповідну фабрику.

Дане архітектурне рішення входить до складу моделі в шаблоні MVC, внаслідок чого користувач втрачає можливість самостійно додати повністю новий компонент у систему чи модернізувати певний компонент. Це можливо лише програмним шляхом.

3.2 **Альтернативні архітектурні рішення**

Для вирішення даної проблеми можна скористатись альтернативним способом і використати породжуючий шаблон Builder.

Шаблон Builder використовується:

— в системі можуть існувати складні об'єкти, створення яких за одну операцію важко або неможливо і необхідна поетапна побудова об'єктів з контролем результатів виконання кожного етапу;

— дані повинні мати кілька предствлень.

Builder відокремлює алгоритм поетапного конструювання складного об'єкта від його зовнішнього уявлення так, що за допомогою одного і того ж алгоритму можна отримувати різні уявлення цього продукту. На рисунку 3.7 показано діаграму класів шаблону Builder.

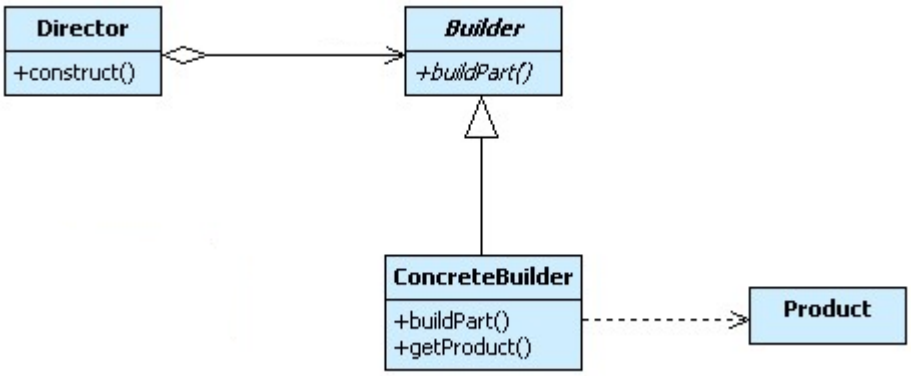


Рисунок 3.7 — UML-діаграма класів шаблону Builder

Класс Director містить вказівник або посилання на Builder, який перед початком роботи повинен бути сконфігурований екземпляром ConcreteBuilder, що визначає відповідне представлення. Після цього Director може обробляти клієнтські запити на створення об'єкта. Отримавши такий запит, за допомогою існуючого екземпляра, Director будує продукт по частинах, а потім повертає його користувачу (рисунок 3.8).

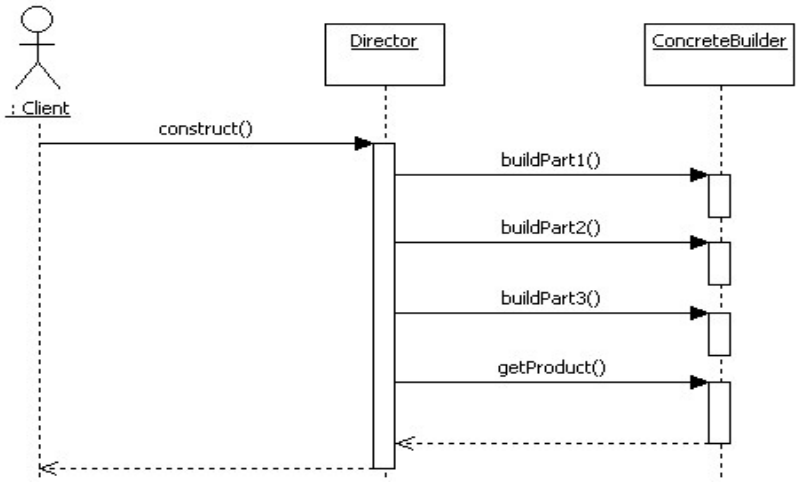


Рисунок 3.8 — UML-діаграма послідовності шаблону Builder

Переваги шаблону Builder:

— можливість контролювати процес створення складної продукції;

— можливість отримання різних представлень деяких даних;

— ізолює код для побудови та представлення. Клієнти нічого не знають про класи, що визначають внутрішню структуру продукту, такі класи не відображаються в інтерфейсі Builder.

Недоліком шаблону є те, що, наприклад, клас ConcreteBuilder і створювана ним продукція жорстко пов'язані між собою, тому при внесенні змін в продукт класу, швидше всього, доведеться відповідним чином змінити і клас ConcreteBuilder.

Ще однією альтернативою шаблону Abstract Method являється шаблон Abstract Factory. Ідея заключається в тому, що для кожного об’єкта створюється відповідний абстрактний метод, реалізацію якого вже було описано в даному розділі. Для того, щоб вся система залишалась незалежною, даний шаблон використовує абстрактний клас, де описаний єдиний інтерфейс, через який можна керувати об’єктами похідних класів.

Різниця між Abstract Factory і Abstract Method полягає в тому, що шаблон Abstract Method являється функцією, а Abstract Factory об’єктом, який може вміщувати декілька абстрактних методів в собі, і залежно від ситуації визивати той чи інший метод. Це дає змогу при розширенні системи, ділити компоненти на сімейства і, наприклад, створювати компонент освітлення з фабрики освітлення.

В даному випадку було вибрано шаблонb Factory Method та Prototyp, тому що:

— система повинна розширятись шляхом додавання об’єктів нових типів;

— заздалегідь відомо, коли потрібно створювати об'єкт, але невідомий його тип;

— створення нового об’єкта відбувається шляхом глибокого копіювання об’єкта базового класу, що дозволяє уникнути дублювання коду.

— створює об'єкти різних типів, дозволяючи системі залишатися незалежною як від самого процесу створення, так і від типів створюваних об'єктів.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі детально описано реалізацію клієнтської і серверної частин, описано структуру бази даних, мережевий протокол та його імплементація в клієнтських та серверних програмах. Також в даному розіділі описано технології, які використовувались для реалізації поставлених цілей.

4.1 Опис клієнтської частини

Програмний продукт складається з двох основних частин: клієнтської і серверної.

Клієнтська частина розроблена на основі архітектурного шаблону MVC та представляє собою три основні взаємодіючих компоненти:

— клієнтський інтерфейс (View);

— сервіси, що забезпечують взаємодію між користувачем та сервером (Controler);

— модуль, який відповідає за створення та модернізацію компонентів (Model).

Зв’язок між об’єктами класів реалізовано за допомогою агрегації та композиції та успадковування [12].

Композиція — це один із видів зв’язку між об’єктами, де один клас не може існувати без іншого. В даному випадку, наприклад, клас House керує часом існування об’єкта DeviceFactory. І об’єкт класу DeviceFactory залежить від класу House і не може існувати окремо.

Агрегація — це один із видів зв’язку між об’єктами, де один клас може існувати незалежно від іншого. Наприклад, в даній реалізації знищення об’єкта класу Model не призведе до знищення об’єкта класу PropertiesFrame, хоча

створення об’єкта класу PropertiesFrame залежить від об’єкта класу Model. На рисунку 4.1 показана структура клієнтської частини [7].

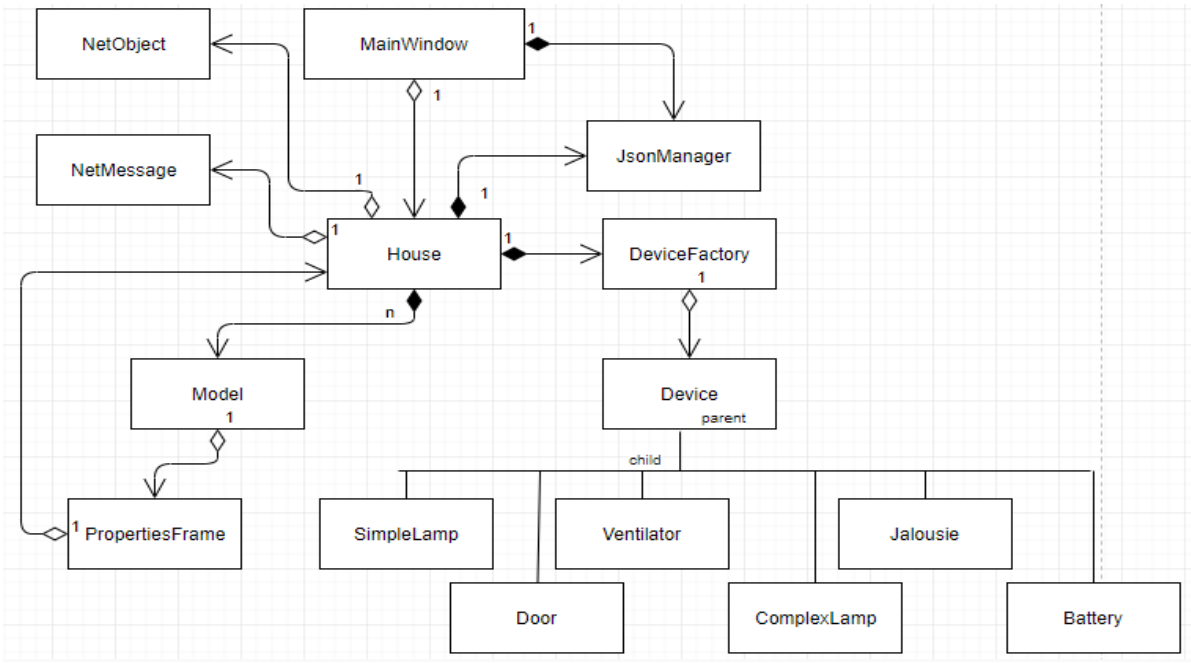


Рисунок 4.1 — Структура клієнтської частини

Клас MainWindow відповідає за створення головного вікна додатка, а також створення та відновлення інтерфейсу користувача. При входженні користувача в систему відбувається відтворення існуючих візуальних компонентів та відображається їх режим роботи.

Клас JsonManager відповідає за зчитування та запис інформації про кожен компонент в json файл, який зберігається на стороні клієнта, та синхронізується з сервером під час початку роботи користувача з додатком.

Клас House являється головним класом для моделі. В ньому містяться всі головні функції для створення, видалення та зміни параметрів компонентів. Даний клас створений на основі шаблону Singleton, який дозволяє створити тільки один об’єкт класу House і використовувати його по всій програмі.

Клас DeviceFactory відповідає за створення об’єкта конкретного класу успадкованого від Device [3]. Після створення, об’єкт типу Device зберігається

в статичному контейнері, ініціалізованому в класі House. Клас Device з усіма успадкованими класами, а також клас DeviceFactory, представляють собою шаблон Factory Method (рисунок 4.2), який вже був описаний в попередньому розділі.

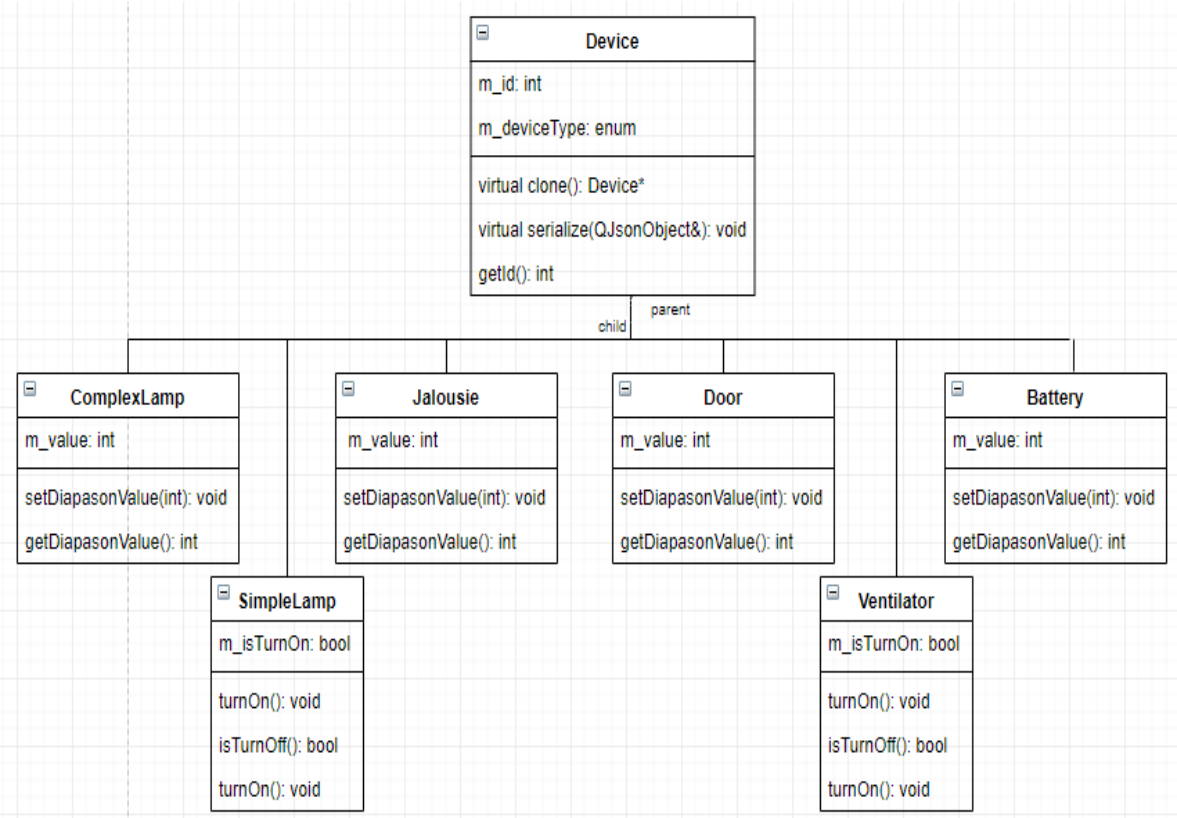


Рисунок 4.2 — Діаграма класів компонентів

В класі Device проініціалізовано віртуальні методи clone, та serialize. Метод clone призначений для створення об’єкта відповідного класу нащадка.
В свою чергу, в методі serialize міститься запис властивостей відповідного об’єкта в json формат.

Кожному об’єкту класу Device створюється відповідний візуальний компонент. Клас Model призначений за створення відповідного графічного компонента і відображення його в конкретному місці в інтерфейсі користувача. Для кожного компонента користувач може викликати меню компонента. PropertiesFrame, яке дозволяє змінити статус роботи пристрою, задати дату та

час зміни роботи пристрою та видалити компонент із системи. Зв’язок між об’єктами класів Model та PropertiesFrame реалізовано через композицію.

Клас NetObject відповідає за пересилання повідомлень між клієнтом та сервером. Передача даних відбувається за допомогою протоколу TCP/IP. Повідомлення представляє собою басив байтів, де містить ідентифікатор повідомлення а також вся необхідна інформація про компонент (ідентифікатор, значення, позиція тощо).

Для створення повідомлень використовується базовий клас NetMessage, в якому ініціалізовано абстрактний метод для перетворення даних в масив байтів. Всі похідні від NetMessage класи перевизначають даний абстрактний метод. На рисунку 4.3 показана діаграма класів, які відповідають за створення повідомлень. Всі похідні класи перевизначають логіку даного методу, дозволяючи вмішувати в повідомлення різні дані.

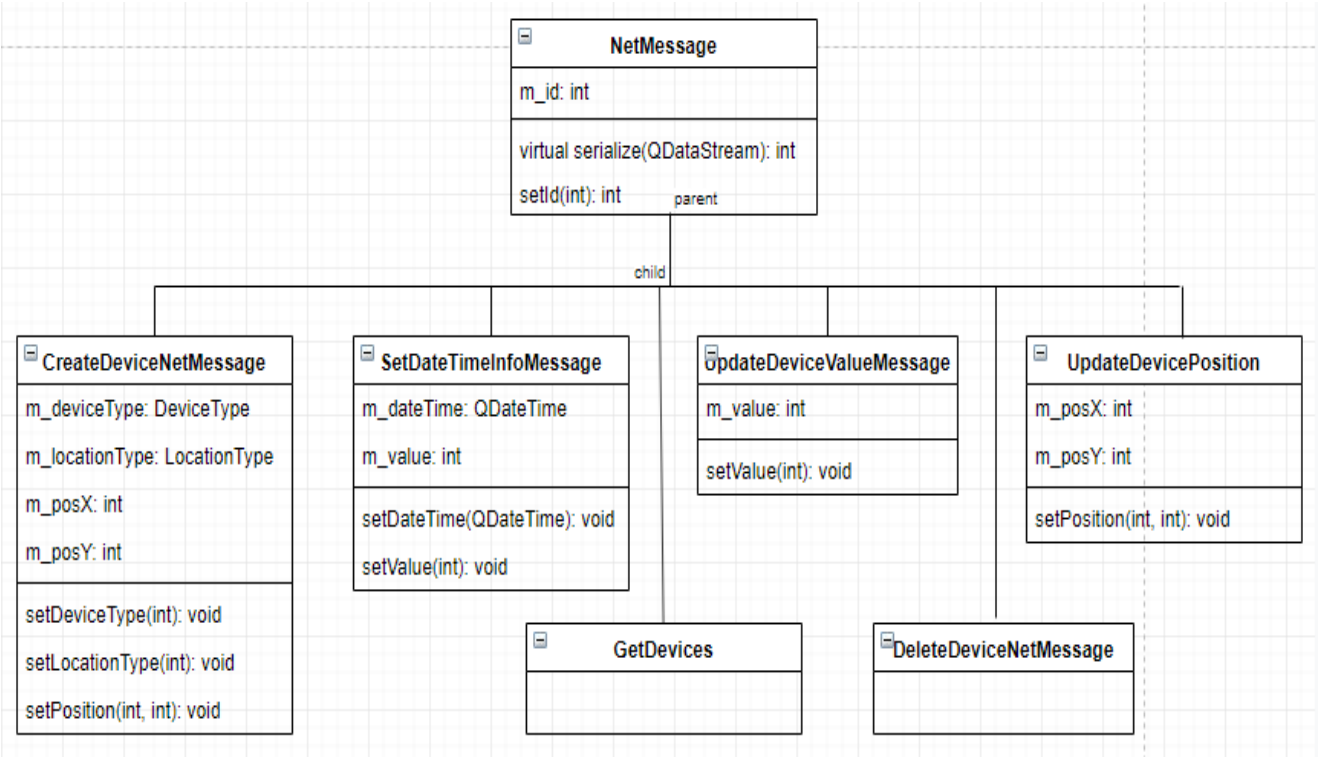


Рисунок 4.3 — Діаграма класів для створення повідомлень

Для визначення типу повідомлення, яке приходить на сервер, використовуються відповідні ідентифікатори повідомлення. Даний

ідентифікатор — це число, яке займає перші чотири байти в масиві (рисунок 4.4). Наступні вісім байтів виділяються для запису ідентифікатора користувача та ідентифікатора візуального компонента. В наступні блоки записується решта інформації, яка залежить від повідомлення, наприклад, нове значення компонента, координати, час тощо.

Рисунок 4.4 — Структура повідомлення

Як на стороні клієнта так і на стороні сервера зареєстровано повідомлення з наступними ідентифікаторами:

— UpdateDeviceStatus — ідентифікує повідомлення про зміну режиму роботи пристрою. Даний тип оголошення додатково містить нове значення пристрою;

— UpdateDevicePosition — ідентифікує повідомлення про оновлення позиції пристрою. В сповіщення додатково записуються нові координати візуального компоненту;

— CreateDevice — ідентифікує повідомлення про створення нового компоненту. Даний тип повідомлення містить координати візуального компонента та ідентифікатор локації;

— DeleteDevice — ідентифікує повідомлення про видалення компоненту;

— GetDevices — ідентифікує повідомлення про отримання інформації всіх компонентів компоненту. В сповіщення даного типу записуються лише ідентифікатор повідомлення та ідентифікатор користувача;
— SetDateTimeEvent — ідентифікує повідомлення про створення часу зміни режиму роботи компонента. В дане сповіщення записується нове значення пристрою і дата та час;
— GetDateTimeEvent — ідентифікує повідомлення про успішне оновлення режиму роботи компонента в певний період часу. В повідомлення записується нове значення пристрою.

4.2 Опис серверної частини

Серверна частина виконана на мові програмування Python і виконує роль проміжного компонента між клієнтом та модулем управління пристроями. Сервер має можливість обмінюватись повідомленнями з клієнтом, модулем керування пристроями та сервером бази даних (рисунок 4.5).

Серверна частина складається з чотирьох основних класів:

— NetObject — клас, який використовується для комунікації з клієнтом та модулем управління пристроями;
— DateTimeManager — клас для управління подіями пов’язані з оновленням роботи компонентів через певний проміжок часу;
— MongoDBManager — клас для формування та відправлення запитів на сервер бази даних;
— DeviceManager — даний клас містить лише інтерфейси, які симулюють поведінку оновлення пристроїв.

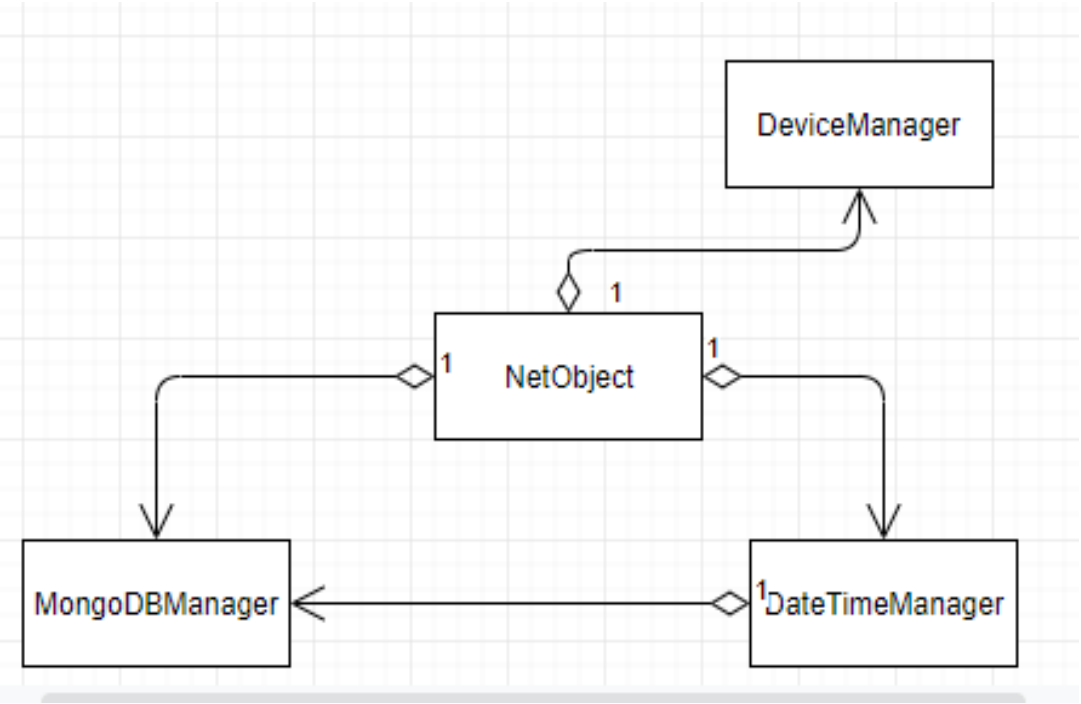


Рисунок 4.5 — Структура серверної частини

Алгоритм обробки повідомлення реалізовано асинхронно і дає змогу одночасно обробляти повідомлення від багатьох користувачів (рисунок 4.6).

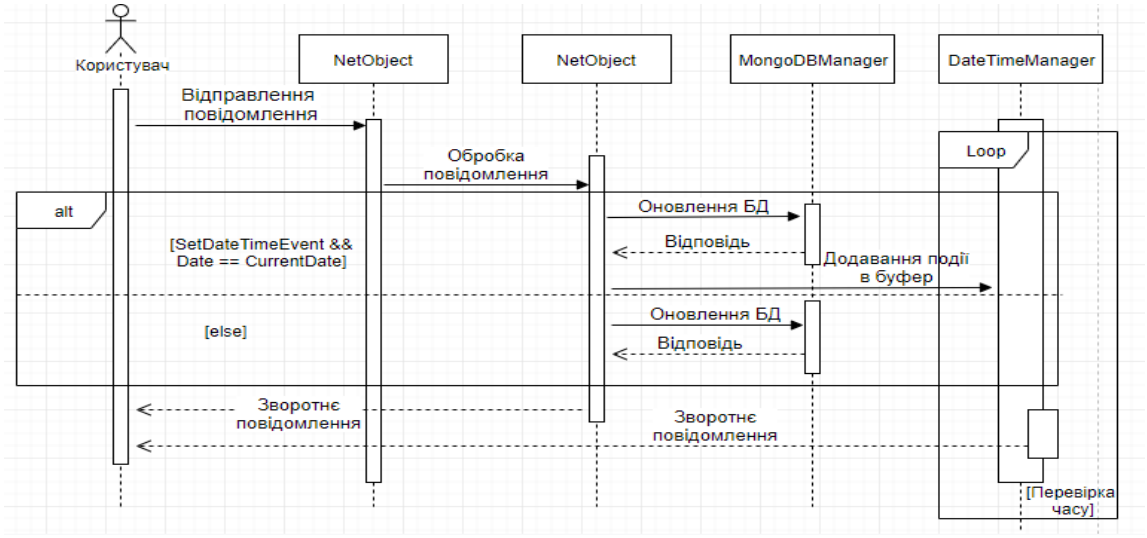


Рисунок 4.6 — Діаграма послідовності обробки повідомлення

Прийом повідомлень винесено в окремий потік з метою одночасного прийому та обробки повідомлень від декількох користувачів. Після прийому повідомлення, відбувається процес зчитування даних з масиву байтів. Перші чотири байти в будь якому повідомленні відведені для ідентифікатора повідомлення. Даний ідентифікатор дає змогу визначити тип повідомлення, та тип даних, які містяться в ньому. Після зчитування формується і відправляється відповідний запит до бази даних. Після завершення зчитування повідомлення, відправляється зворотнє сповіщення користувачу з наступним кодом:

— Success — повідомлення з даних типом відправляється у разі успішної обробки повідомлення;
— InvalidMessageType — повідомлення з даних типом відправляється якщо сервер не зміг ідентифікувати вхідне сповіщення;
— InvalidDeviceParameters — повідомлення з даних типом відправляється якщо порушена структура вхідного сповіщення;
— InvalidUserId — повідомлення з даних типом відправляється якщо не вдалося ідентифікувати користувача.

Клієнт має можливість задати час та дату оновлення режиму роботи (включення чи виключення) кожного компонента. Процес моніторингу часу також виведено в окремий потік. Дані про зміну роботи пристрою через певний період часу зберігаються в окремій таблиці бази даних. В кінці кожної доби сервер зчитує з бази даних всі записи, які повинні бути оброблені на протязі наступної доби і заносить їх в список очікування. На початку нової ітерації відбувається перевірка наявності даних в списку очікування. Якщо дані існують, то вони переміщуються зі списку очікування в активний список. Принцип моніторингу полягає в тому, щоб перевірити час кожного елемента в активному списку і якщо відповідний час настав, то оновити інформацію про компонент в базі даних і надіслати повідомлення користувачу та модулю керування пристроями. Використанням списку очікування та активного списку

дозволяє уникнути неочікуваної поведінки програми та вихід за межі масиву. Ідея полягає в тому, що доки потік обробляє активний список, в список очікування приходить нова інформація і перед новою обробкою активного списку відбувається переміщення даних з тимчасового списку. Таким чином в момент обробки активного списку його розмір не збільшується.

Як на стороні клієнта, так і на стороні сервера реалізовано обробку втрати зв’язку, щоб уникнути неочікуваного завершення програм.

На стороні клієнта, у випадку втрати зв’язку з сервером викликається сигнал disconnected. Слот, що реагує на даний сигнал реалізовано у вигляді лямбда функції, яка сповіщає користувача про втрату зв’язку, видаляє сокет і запускається процес відновлення зв’язку з сервером.

На стороні сервера реалізовано блок try finally, щоб уникнути завершенню програми. Блок try відстежує втрату зв’язку або іншої неочікуваної помилки і в разі її виявленні викликається блок finally, де даний користувач видаляється зі списку активний користувачів і сокет закривається.

Під час відправлення повідомлення на сервер, дане повідомлення реєструється у контейнері у вигляді пари ключ значення, де ключем являється ідентифікатор повідомлення, а значенням ідентифікатор компонента. Якщо повідомлення було успішно оброблене, сервер надсилає користувачеві повідомлення про успішну обробку і на стороні клієнта, повідомлення, яке було відправлене видаляється з реєстру. Якщо клієнт отримав інформацію про помилку, то повідомлення формується і відправляється на сервер повторно.

4.3 Структура бази даних

В якості СКДБ було використано MongoDB [7]. Дана СКДБ не потребує опису схем різних таблиць і ідеально підходить для роботи з даними, які не містять відношень. Всі дані зберігаються у вигляді документів у двох

колекціях: devices і datetime_requests. В колекції devices зберігаються інформація про візуальні компоненти (рисунок 4.7). Колекція datetime_requests містить всі події про оновлення компонентів в певний період дати та часу (рисунок 4.8). Під час додавання нового запису автоматично генерується унікальний ідентифікатор (ObjectId).

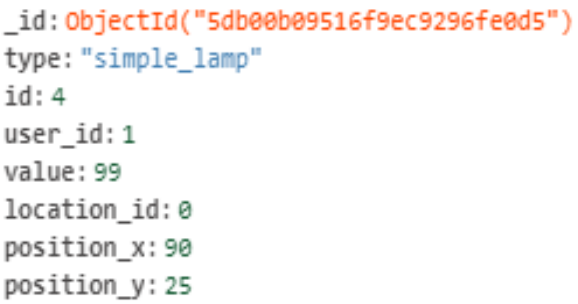


Рисунок 4.7 — Документ колекції devices, який містить дані візуального компонента

Рисунок 4.8 — Документ колекції datetime_requests, який містить дані для оновлення значення компонента в певний період часу

Для роботи з даним СКДБ використовувався модуль pymongo, який містить клас MongoClient. Даний клас містить методи для встановлення зв’язку з сервером бази даних, а також методи для відправлення запитів. Для формування запитів до бази даних використовувались наступні методи:

— insert_one — використовувався для вставки даних кожного компонента в базу даних;

— update_one — використовувався для оновлення властивостей компонента;

— delete_one — відповідає за видалення компонента з бази даних;

— find — відповідає за пошук компонентів.

4.4 Опис реалізації взаємодії клієнта і сервера

Для передачі даних між додатком користувача та сервером використовувався протокол TCP/IP [5]. На стороні клієнта, всі алгоритми для встановлення зв’язку з сервером, прийом та відправка повідомлень реалізовувались за допомогою бібліотеки QtNetwork. На стороні сервера даний функціонал реалізовувався за допомогою модулю socket. Обмін даними між клієнтом там сервером відбувається завдяки сокетам [10].

TCP/IP (Протокол керування передачею) — це надійний транспортний протокол, орієнтований на потоки, вміщує в собі прокол TCP та протокол IP. Це особливо добре підходить для безперервної передачі даних. Основна перевага даного протоколу полягає в тому, що він забезпечує надійну доставку даних. На рисунку 4.9 показано схема встановлення зв’язку між клієнтом і сервером.

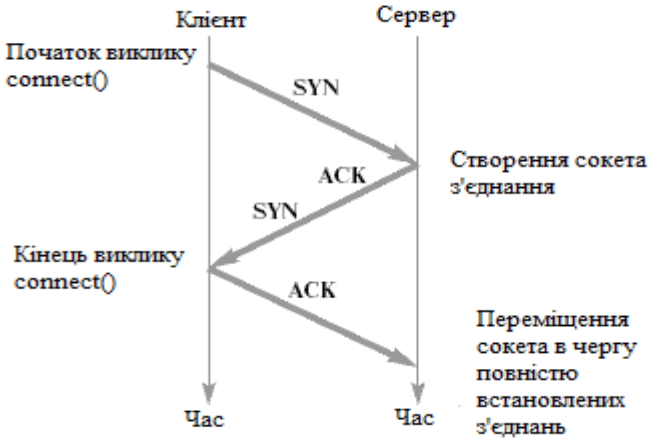


Рисунок 4.9 — Процес встановлення TCP/IP з’єднання

З’єднання відбувається в три етапи:

— відправлення сокета на сервер;

— прийом сокета та відправлення інформації клієнту про успішне приєднання;

— прийом сигналу від клієнта і можливість передачі даних.

Мережевий сокет — це внутрішня кінцева точка для відправлення або отримання даних на одному вузлі в комп’ютерній мережі. Слід розрізняти клієнтські і серверні сокети. Клієнтські сокети грубо можна порівняти з кінцевими апаратами телефонної мережі, а серверні з комутаторами. Відповідно клієнтський додаток використовує лише клієнтські сокети, а серверний як клієнтські так і серверні сокети.

Мережеві сокети більш детально поділяються на:

— датаграм сокет — це тип мережевого сокету, який забезпечує точку з'єднання для надсилання або отримання пакетів даних. Порядок та надійність не гарантуються, тому декілька пакетів, відправлених з однієї машини чи процесу на інший, можуть надходити в будь-якому замовленні або взагалі не надходити;

— потоковий сокет — це тип мережевого сокету, який забезпечує орієнтований на зв'язок, послідовність та унікальний потік даних без меж запису, з чітко визначеними механізмами створення та знищення підключень та виявлення помилок. Даний тип сокетів використовувався в для передачі повідомлень між клієнтом та сервером;

— сирий сокет — це мережевий сокет, який дозволяє здійснювати пряму передачу та прийом IP-пакетів без будь-якого форматування транспортного рівня, специфічного для протоколу.

Що стосується інших видів сокетів, то корисне навантаження автоматично приховується відповідно до вибраного протоколу транспортного

рівня (наприклад, TCP, UDP), і користувач сокету не знає про існування заголовків протоколів, які транслюються з корисним завантаженням. При читанні з сирого сокету заголовки зазвичай включаються. Під час передачі пакетів із сирого сокету автоматичне додавання заголовка є необов'язковим.

Для того щоб клієнт міг взаємодіяти з сервером, йому потрібно знати його IP-адресу і номер порту, через який клієнт повинен повідомити про себе. Коли клієнт встановлює з'єднання з сервером, система призначає даному з'єднанню окремий сокет. Після цього встановлюється зв'язок між двома даними сокетами, по якому надсилаються дані запиту до сервера. А сервер висилає клієнту, по цьому з'єднанню, готові, оброблені результати згідно його запитам. Сервер не обмежений зв'язком тільки з одним клієнтом, насправді він може обслуговувати багатьох клієнтів. Кожному сокету відповідає унікальний номер порту. Деякі номери портів зарезервовані для так званих стандартних служб.

Переваги протоколу TCP/IP [13]:

—це популярний стек протоколів;

—відзначається надійною доставкою пакетів та правильним порядком прийому даних;

—усі сучасні операційні системи підтримують стек TCP/IP;

—стек TCP/IP є основою гнучкої технології для поєднання різнорідних систем і мереж, як на рівні реалізації транспортної функції, так і на рівні взаємодії прикладних процесів. Стек TCP/IP забезпечує масштабоване середовище для застосовань клієнт-сервер.

Недоліки протоколу TCP/IP:

—відсутність розмежувань концептуальних понять інтерфейсу, протоколу та рівневого сервісу, що досить чітко зроблено в моделі ISO/OSI. Внаслідок цього модель TCP/IP не може застосовуватися для розробки нових мереж;

— за допомогою моделі TCP/IP неможливо описати жоден інший стек протоколів, окрім TCP/IP;

— у моделі не диференційовано фізичний та канальний рівні, хоча вони абсолютно різні, що в коректній моделі обов’язково враховується;

— швидкість доставки повідомлення менша в порівнянні з протоколом UDP;

— стек TCP/IP не може розглядатися як повноцінна модель, однак самі протоколи добре протестовані та надзвичайно популярні.

Серверний сокет створювався з використання модуля socket, який дає можливість налаштувати сокет у вигляді об’єкта класу socket. Для ініціалізації сокета були виконані наступні дії:

— створено сокет з адресним сімейством типу AF_INET та типом SOCK_STREAM. AF_INET тип адреси, який використовується для мережевого протоколу IPv4. Тип SOCK_STREAM вказує, що даний сокет являється потоковим;

— приєднано сокет до IP-адреси та порту;

— встановлено сокет в режим прослуховування. В даному режимі сокет перевіряє порт та очікує на приєднання користувачів.

Для ініціалізації сокета користувача в клієнтському додатку використовувався клас QTcpSocket.

Клас QTcpSocket — це клас, успадкований від базового класу QAbstractSocket, який дозволяє встановлювати TCP-з'єднання та передавати потоки даних. Перед передачею даних може бути встановлено зв'язок TCP із віддаленим хостом і портом. QTcpSocket працює асинхронно та випускає сигнали для повідомлення змін статусу та помилок. Для обробки сигналів прийому повідомлення та втрати зв’язку були створені відповідні слоти (методи).

4.5 Переваги реалізації клієнтської частини в середовищі Qt

Qt Creator являє собою інтегроване середовище розробки, призначене для створення крос-платформових додатків з використанням бібліотеки Qt. Підтримується розробка як класичних програм мовою C++, так і використання мови QML, для визначення сценаріїв, в яких використовується JavaScript, а структура і параметри елементів інтерфейсу задаються CSS-подібними блоками. Для Windows версій бібліотека комплектується компілятором, заголовними і об'єктними файлами MinGW. Додаток користувача реалізований мовою програмування C++ на основі фреймворку Qt 5, який вийшов у грудні 2012 та примітний модульною структурою. Незважаючи на багато істотних поліпшень і змін, Qt 5 зберігає базову зворотну сумісність із минулими випусками, підтримує повною мірою засоби для створення Qt-програм мовою C++ і містить майже всі компоненти Qt 4.

Відмінна особливість Qt від інших бібліотек виявляється в тому, що тут використовується Meta Object Compiller (MOC) — компілятор, який дозволяє динамічно обробляти інформацію в реальному часі та механізм сигналів і слотів. Загалом, Qt — це бібліотека не для чистого C++, а для його особливого діалекту, який обробляється компілятором MOC для подальшої компіляції будь-яким стандартним C++ компілятором. MOC дає змогу в багато разів збільшити потужність бібліотек, вводячи такі поняття, як слоти (slots) і сигнали (signals). Qt комплектується графічним середовищем розробки графічного інтерфейсу QTDesigner, що дозволяє створювати діалоги і форми. Ідеологія створення форм у Qt базується на використанні менеджерів розташування, котрі надають гнучкий дизайн, при якому розмір і розташування елементів форм визначаються автоматично, що значно прискорює розробку графічного інтерфейсу. Даний фреймворк також відомий завдяки інструментам Qt Linguist — могутня

графічна утиліта, що дозволяє спростити локалізацію й переклад вашої програми багатьма мовами, та Qt Assistant — довідкова система Qt, що спрощує роботу з документацією для бібліотек і дозволяє створювати крос-платформову довідку для програмного забезпечення, розробленого на основі Qt.

В Qt Creator вбудований Qt Designer для розробки графічних додатків і довідкова система Qt Assistant, QML дизайнер. В Qt Creator є можливість розробки на мовах C++, JavaScript, QML. Вся бібліотека Qt розділена на модулі і дає можливість працювати з різними компонентами програмування такими як: бази даних, мережі, тести, xml, opengl. Qt Creator дає можливість компілювати код з використанням різних компіляторів. В даному проекті використовувався Qt 5 та компілятор MinGW 64. При розробці даної програми були використані всі новітні стандарти мови C++ [9].

В даній програмі активно використовувались технологій сигналів та слотів для комунікації із сервером та обробки вхідної інформації користувача [14]. Сигнали і слоти активно використовуються для комунікації між об’єктами, наприклад, при активації сигналу натискання клавіш миші виконується певний алгоритм, тобто сигнал виробляється коли відбувається якась подія.

Слот — це функція, яка визивається як відповідь на якийсь сигнал. Віджети Qt мають безліч визначений сигналів і слотів, що дозволяє створити свої сигнали чи слоти. У даному механізмі сигнатура сигналу повинна збігатися з сигнатурою слота-одержувача. В певних випадках слот може мати більш коротку сигнатуру ніж сигнал який він отримує, так як він може ігнорувати додаткові аргументи. Сигнатури сигналів та слотів порівнюються в реальному часі, а не на стадії компіляції. Сигнали і слоти слабо пов'язані, тому клас, який виробляє сигнал не знає і не піклується про те, які слоти його обробляють. Механізм сигналів і слотів гарантує, виклик відповідного слоту в потрібний час у разі, якщо параметри сигналу будуть співпадати з сигналами слоту . Сигнали і слоти можуть приймати будь-яке число аргументів будь-якого типу. Всі класи,

успадковані від QObject або від похідних від QObject класів, можуть містити власні сигнали і слоти . Сигнали виробляються об'єктами коли вони змінюють свій стан. Так само як об'єкт не знає нічого про одержувачів своїх сигналів, слот нічого не знає про сигнали, які до нього підключені. Це дозволяє створювати проекти, де компоненти не тісно залежать один від одного, що надає певної гнучкості.

Зв’язування об’єктів та слотів відбувається за допомогою функції connect, яка міститься в класі QObject. Метод connect Сигнатура метода connect зображена на рисунок 4.10.

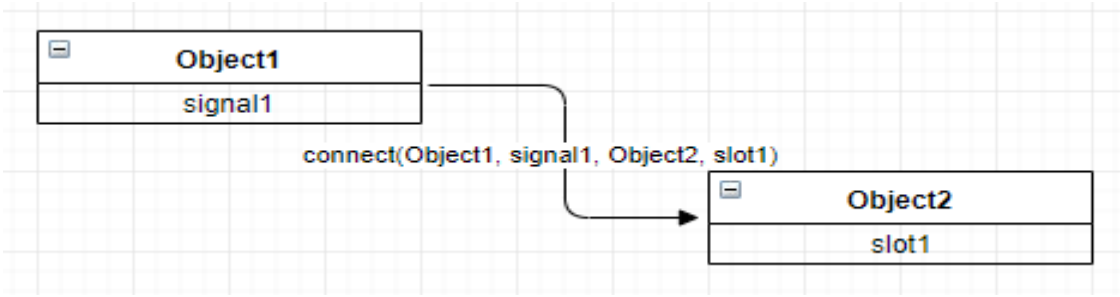


Рисунок 4.10 — Сигнатура метода connect.

Об’єкт Object1 є відправником сигналу. Даний об’єкт приймає сигнал якоїсь події і форму сигнал певного типу, наприклад, об’єкт класу QTcpSocket приймає повідомлення від сервера та формує сигнал readyRead. Тип сигналу записується на місті сигналу signal1. Об’єкт класу QObject2 являється об’єктом, який приймає сигнал. Слот slot1 являється методом, з вхідними параметрами, який буде визиватись внаслідок спрацювання сигналу. Параметри сигналу повинні обов’язково співпадати з параметрами слота.

Для обробки інформації користувача (натискання клавіш миші, переміщення курсору тощо) використовувались події. У Qt події є об'єктами, похідними від абстрактного класу QEvent, які репрезентують речі, які сталися як в межах програми, так і внаслідок зовнішньої діяльності, яку програмі потрібно знати. Події можуть бути отримані та оброблені будь-яким об’єктом

підкласу QObject, але вони особливо актуальні для віджетів. Коли відбувається подія, створюється об'єкт події і доставляється до певного екземпляра QObject (або одного з його підкласів), викликаючи функцію event. Ця функція не обробляє саму подію, а викликає обробника події, який підписаний на даний тип події і надсилає відповідь, залежно від того, була ця подія прийнята чи проігнорована. Деякі події, такі як QMouseEvent та QKeyEvent, надходять з вікна системи, інші, такі як QTimerEvent, надходять з інших джерел чи від самої програми.

Клас QMouseEvent містить параметри, що описують події миші. Події, пов'язані з мишею, активуються, коли кнопка миші натискається чи відпускається всередині віджету або коли рухається курсор миші. Система автоматично відстежує мишу, коли відповідна кнопка натискається. Віджет буде продовжувати отримувати події миші, доки не буде відпущена відповідна кнопка. Подія для миші містить спеціальний прапорець, який вказує на те, чи бажає одержувач обробляти її. Команда ignore ігнорує подію миші, а accept навпаки приймає Функції pos, x і y дають позицію курсору відносно віджета, який отримує подію миші.

В додатку користувача були використані події типу QMouseEvent, а саме:

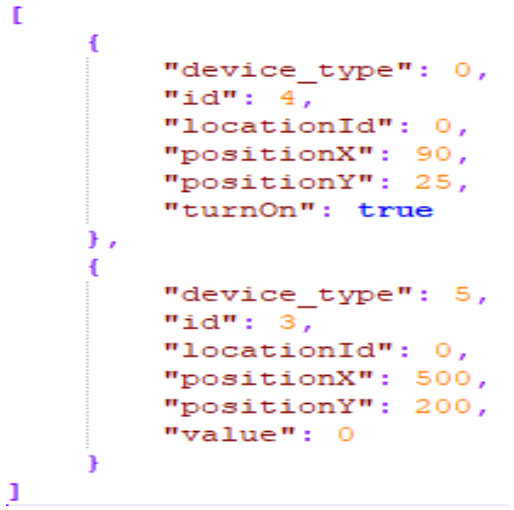
—mouseMoveEvent — для відстежування переміщення курсору миші;

—mousePressEvent — для відстежування натискання правої кнопки миші;

—mouseDoubleClickEvent — для відстежування подвійного натискання лівої кнопки миші.

Система подій тісно пов’язана із системою сигналів та слотів. Коли спрацьовує певна подія, формується сингал та викликається, який містить вказівник на дану подію. Внаслідок спрацювання сигналу, викликається відповідний слот, де відбувається обробка даної події. Наприклад, під час натискання правої кнопки миші, в слоті відбувається створення та відображення додаткового меню для віджета.

При створенні, видаленні чи зміні стану кожного пристрою відбувається оновлення json файлу (рисунок 4.11), куди заноситься повна інформація про пристрій. Дане рішення дає можливість навіть без інтернет зв'язку візуально надати користувачеві інформацію про кожен пристрій, наприклад, працює даний пристрій чи ні. Кожен компонент у файлі виступає як окремий блок, де міститься інформація про компонент, а саме: тип компоненту, ідентифікатор, ідентифікатор локації, координати на робочому плані і значення. Всі компоненти вкладені в масив.



```
[
    {
        "device_type": 0,
        "id": 4,
        "locationId": 0,
        "positionX": 90,
        "positionY": 25,
        "turnOn": true
    },
    {
        "device_type": 5,
        "id": 3,
        "locationId": 0,
        "positionX": 500,
        "positionY": 200,
        "value": 0
    }
]
```


Рисунок 4.11 — Структура json файлу для візуальних компонентів системи

Для зчитування та запису даних з json файлу використовувався клас QFile. В даному класі містяться відповідно методи readAll та write, куди передається у якості параметра об'єкт класу QJsonDocument. Об'єкт QJsonDocument дозволяє записати дані у масив. Інформація про кожен візуальний компонент записується в об'єкт класу QJsonObject. В даному класі містить метод insert, який записує дані у вигляді ключ значення. В батьківському класі Device міститься віртуальний метод serialize, який записує ідентифікатор та тип компонента в об'єкт класу QJsonObject. Всі похідні від Device класи, переініціалізовують метод serialize, добавляючи інформацію про значення компонента (діаграма класів компонентів була описана вище в розділі). Також метод serialize

описується в класі Model, де в об’єкт класу QJsonObject дописуються координати компонента та ідентифікатор локації. Об’єкти класу QJsonObject дадаються в об’єкт класу QJsonArray, який в свою чергу додається в об’єкт класу QJsonDocument, як було зазначено вище. Процес зчитування даних відбувається навпаки, з об’єкта класу QJsonDocument дістається масив об’єктів класу QJsonObject і за зчитаною інформацією створюються відповідні об’єкти класів Device і Model.

4.6 Розробка GUI

Графічний інтерфейс був розроблений завдяки Qt Designer — інструменту розробки графічного інтерфейсу GUI на бібліотеці Qt [6]. Для побудови графічного інтерфейсу для даного продукту використовувались віджети. Кожен віджет представлений персональним класом, який унаслідується від базового класу QWidget. Клас QWidget — це базовий клас усіх об'єктів інтерфейсу користувача.

Віджет (блок) — це одиниця користувальницького інтерфейсу: він отримує сигнал миші, клавіатури та інші події з вікна системи. Кожному блоку відповідає певне зображення на екрані. Даний дизайнер дозволяє створити композитні віджети. Композитні віджети — це віджети, які можуть в собі містити набір інших вбудованих блоків, які сортуються в певному порядку. Багатий набір інструментів дозволяє налаштувати різні властивості віджетів (колір фону, розмір, положення тощо). Для створення графічних вікон використовувались класи QMainWindow та QFrame, які успадковуються від базового класу QWidget.

Для створення інтерфейсу користувача використовувався композитний віджет QTabWidget. Клас QTabWidget дозволяє створити віджет із вкладоками, кожна з яких являється композитним блоком. Віджет із вкладками — це панель

вкладок, яка використовується для відображення сторінок, пов'язаних із кожною з вкладок. За замовчуванням панель вкладок відображається над областю сторінки, але доступні різні конфігурації. Кожна вкладка асоціюється з окремим віджетом (сторінкою). Тільки поточна сторінка відображається в області сторінок, всі інші сторінки приховані. Користувач може відобразити окрему сторінку, натиснувши по відповідній вкладці. Qt Designer також дозволяє додати комбінацію клавіш для відображення певної сторінки.

Для відображення тексту і картинок використовується блок візуалізації. Віджет QLabel дозволяє відображати текст або малюноки. Даний клас не підтримує ніяких функцій взаємодії з користувачем. Візуальну поведінку даного блоку може бути задано різними способами. В проекті даний віджет використовувався для відображення тексту та графічного представлення компонентів у вигляді картинок. Даний віджет можна повністю налагоджувати, задаючи фон, форму рамки, розмір тексту, позицію на екрані тощо.

Для підтвердження певних операцій використовувались кнопки. Віджет QPushButton представляє собою кнопку, яка кивликає певний сигнал сигнал. В даному проекті всі кнопки викликаються сигнал clicked, для якого описані відповідні слоти. Також даний віджет дозволяє викликати сигнали:

— preseed — для відстеження утримання натиснутої кнопки;

— released — для відстеження відпускання натиснутої кнопки;

— toggled — для реалізації перемикання.

Віджет QSlider дає змогу створити вертикальний або горизонтальний повзунок. QSlider — це віджет для задання певного значення у певному діапазоні. Переміщуючи прапорець слайдера вздовж горизонтальної або вертикальної канавки, формується ціле значення, яке відповідає позиції даного прапорця. В додатку користувача, даний віджет використовується для встановлення значення компонента. Клас QSlider містить функції для задання конкретного значення та дозволяє змінювати крок зміщення прапорця. В

додатку реалізовано слот onChangeDiapason, який реагує на зміну положення прапорця та оновлює значення компонента. Крок зміщення залежить від пристрою і для простих компонентів від дорівнює сто, а для всіх інших компонентів значення дорівнює одиниці. Метод setPageStep дозволяє задати крок зміщення. Недоліком є те, що клас QSlider забезпечує лише цілі діапазони значень.

Для створення віджета встановлення часу, використовувався клас QTimeEdit. Даний клас дає змогу встановити зручний формат відображення часу, що дозволяє на вибір користуватись як дванадцяти годинним форматом, так і двадцяти чотирьох годинним форматом. В додатку клієнта час представляється у вигляді рядка символів у форматі “HH:mm:ss”, Даний формат представляє години у двадцяти чотирьох годинному, завжди двоцифровому стилі, хвилини та секунди також представляються у двоцифровому вигляді.

Для комфортного становлення дати, користувачеві представлений календар. Віджет календаря створюється за допомогою класу QCalendarWidget. Так само як і з часом, дата представляється у форматі “dd.MM.yy”. Даний формат дозволяє відображати дні та місяці у двоцифровому вигляді, рік відображається як чотирьох цифрове число.

4.7 Тестування

Для тестування функціоналу на стороні сервера, було використано модуль unittest. Для тестування потрібно створити клас, який успадковується від класу TestCase, що міститься в модулі unittest. Тестовий метод повинен починатися зі слова “test”, і система ідентифікує даний метод на етапі ініціалізації та викличе його на етапі виконання тестування.

Перед виконанням кожного тесту викликається метод setUp, де відбувається створення всіх необхідних ресурсі для тестування, наприклад, в

даному проекті було створено об’єкт класу NetObject.

Після завершення тестування викликається метод tearDown, де відбувається процес очищення всіх ресурсів, які були створені на початку. В даній програмі видаляються всі об’єкти, які були створені в методі setUp.

На рисунку 4.12 зображено приклад виконання тестів.

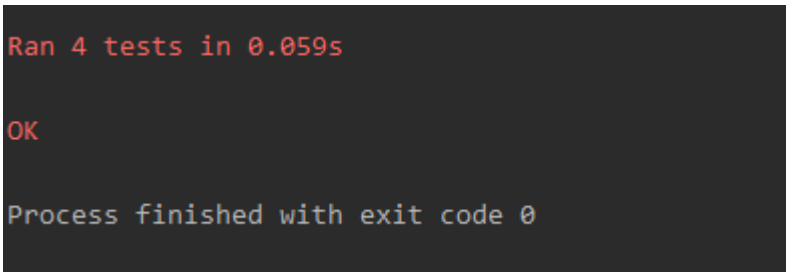


Рисунок 4.12 — Результати виконання тестів

Для тестування додатку користувача, використовувалась бібліотека QTest. Дана бібліотека використовується для написання юніт тестів. Для створення тестів необхідно створити тестовий клас, який успадковується від класу QObject і додати в приватний блок слоти, які і являються тестовими функціями. Також можна додати наступні слоти, які не будуть проестовані, але будуть використані для ініціалізації та очищення даних:

— initTestCase — даний метод буде викликаний перед першим тестом. Використовується для ініціалізації ресурсів, які необхідні всім тестам;
— cleanupTestCase — даний метод буде викликаний після завершення виконання останнього тесту. Використовується для очищення всіх ресурсів, які необхідні були всім тестам;
— init — даний метод буде викликаний перед кожним тестом. Також використовується для ініціалізації ресурсів для тесту;
— cleanup — даний метод буде викликаний після завершення кожного тесту для чищення ресурсів.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ІНСТРУМЕНТАЛЬНИМИ ЗАСОБАМИ МОДЕЛЮВАННЯ СИСТЕМ АВТОМАТИЗАЦІЇ СПОРУД

Весь інтерфейс користувача представлений у вигляді плану будинку. Користувач повинен завантажити фото з планом будинку в програму (рисунок 5.1).

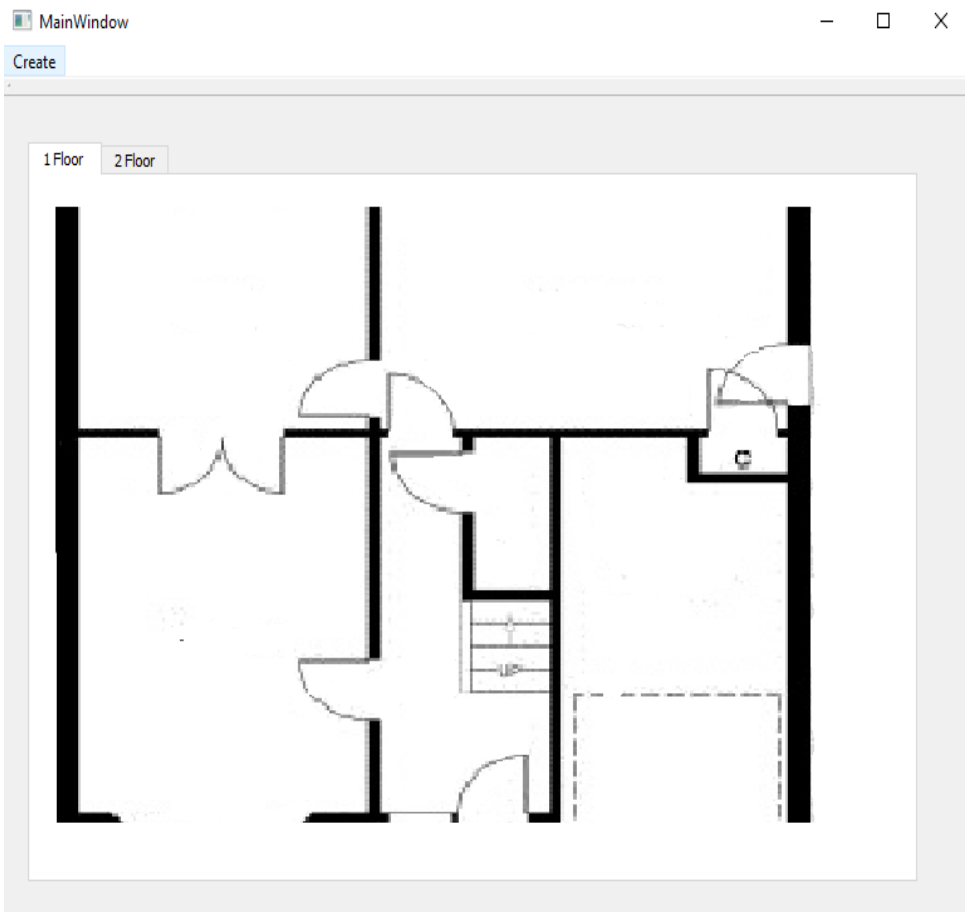


Рисунок 5.1 — План будинку

План представляє собою картинку, яка відображається за допомогою віджета Qlabel, який вкладений у віджет QTabWidget. Віджети всіх створених користувачем компонентів прикріплені до базового віджета QTabWidget, що дає змогу відображати тільки ті компоненти, пристрої які розміщені на певному плані. Зміна планів відбувається за допомогою вкладок. Після запуску програми за замовчуванням відображається завантажений план і відбувається процес підключення до сервера. У разі виникнення помилки підключення з'являється інформаційне вікно, яке повідомляє користувачу про помилку(рисунок 5.2).

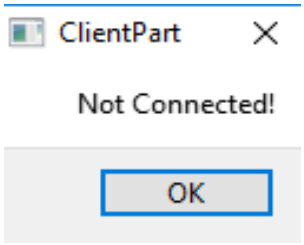


Рисунок 5.2 — Помилка підключення

Після завершення запуску програми відбувається зчитування json файла та відбувається синхронізація даних з сервером, що дає змогу відтворити створені компоненти на плані будинку. Якщо дані сервера не співпадають з даними користувача, то користувачеві будуть присвоєні дані сервера. Навіть у разі помилки підключення до сервера, користувач має можливість переглянути інформацію конкретного пристрою, але дана інформація може бути не правдива.

Разом із планом завантажуються всі створені візуальні компоненти, відображаючи стан відповідного пристрою. Користувач також може добавляти нові компоненти із запропонованого списку (рисунок 5.3).

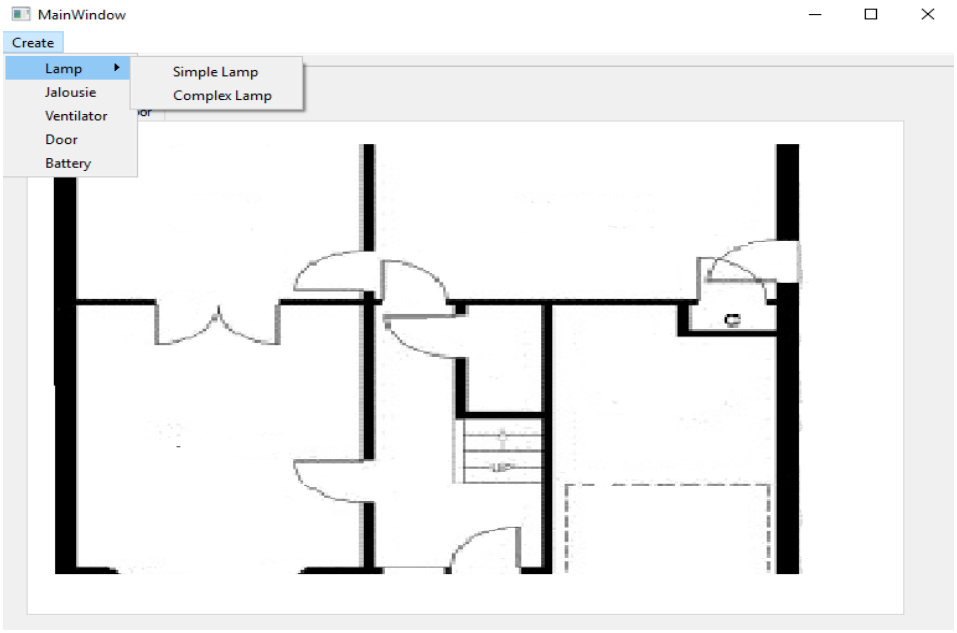


Рисунок 5.3 — Каталог візуальних компонентів

Архітектура програми розроблялась таким, чином, щоб компоненти ділились на різні групи і користувач міг вибирати відповідний компонент з відповідної групи. Наприклад, всі компоненти, які відповідають за освітлювальні пристрої, належать до групи освітлення.

Створення компонента відбувається за допомогою панелі вкладок, де спливає меню із списком доступних компонентів. Вибравши компонент, створюється об’єкт та його візуальна модель, яку користувач може переміщувати в конкретне місце на плані (деталі створення описані в попередніх розділах).

Візуальний компонент відображає робочий стан пристрою. Якщо іконка не підсвічується жовтим кольором, значить пристрій вимкнений (рисунок 5.4), в іншому випадку даний пристрій увімкнений (рисунок 5.5).

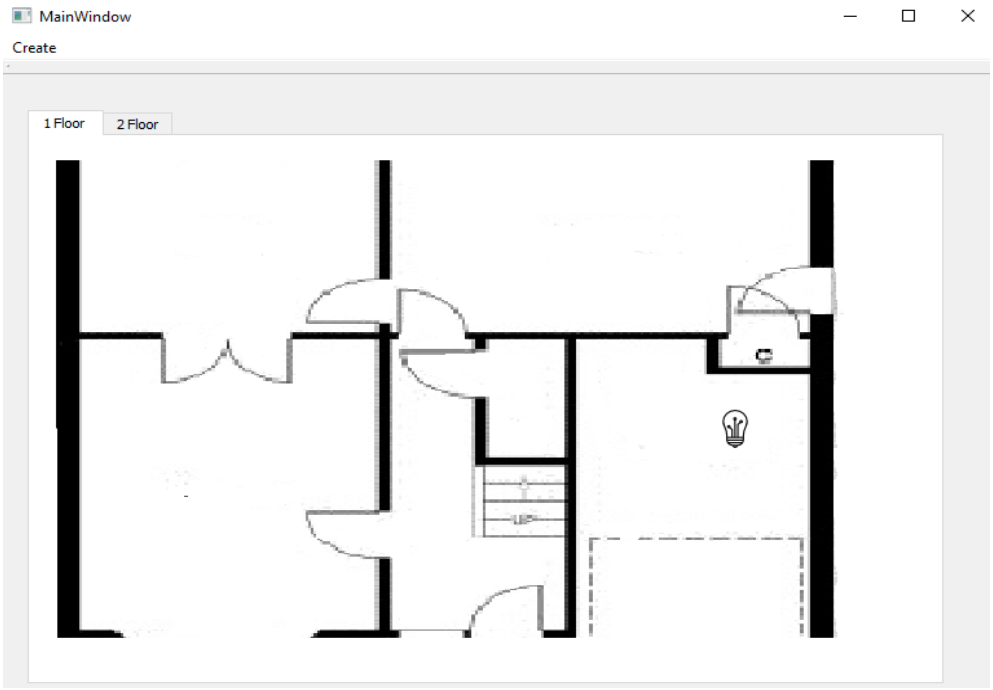


Рисунок 5.4 — Пристрій неактивний

В даному випадку візуальний компонент відображає пристрій освітлення, який перебуває у вимкненому стані.

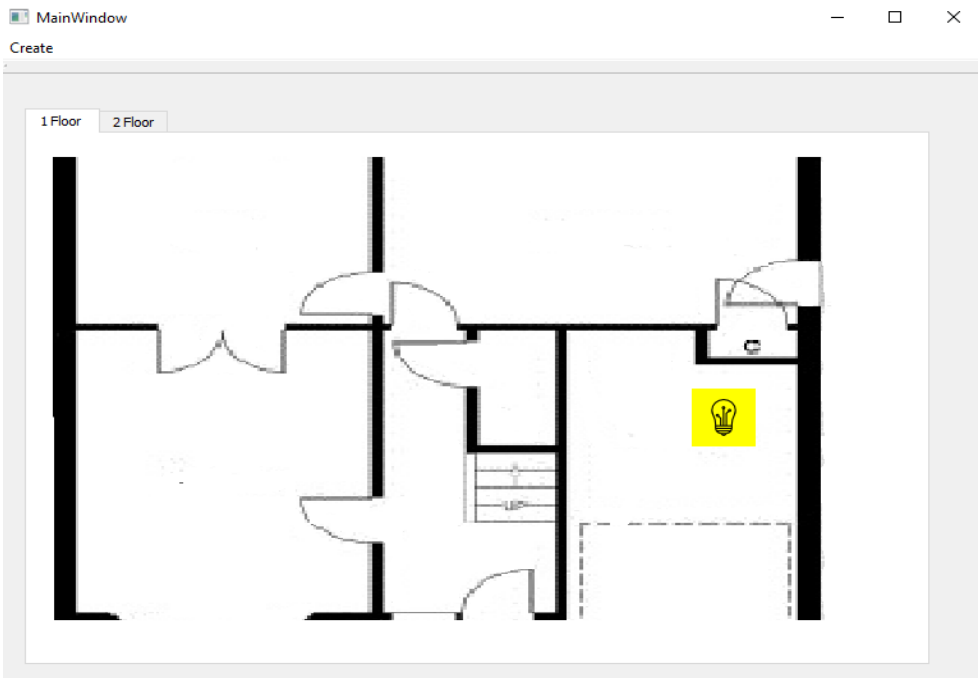


Рисунок 5.5 — Пристрій активний

В даній програмі користувач може створити шість різних візуальних компонентів, серед них:

— компонент простої лампи — надає змогу користувачеві увімкнути або вимкнути освітлювальний пристрій;
— компонент складної лампи — надає змогу користувачеві увімкнути, вимкнути освітлювальний пристрій або регулювати яскравість освітлення;
— компонент вентилятора — надає змогу користувачеві увімкнути або вимкнути вентилятор;
— компонент дверей — надає можливість користувачеві керувати дверима, відкриваючи або закриваючи їх повністю або на певний діапазон;
— компонент жалюзі — надає змогу регулювати позицію жалюзі, піднімаючи чи опускаючи їх на певний діапазон;
— компонент нагрівача — надає змогу керувати системою опалення у приміщенні.

При натисненні правої клавіш миші на візуальний компонент з’являється меню пристрою (рисунок 5.6), де користувач може вибрати опцію видалення даного компонента, задати режим роботи або виставити дату та час зміни режиму роботи пристрою.

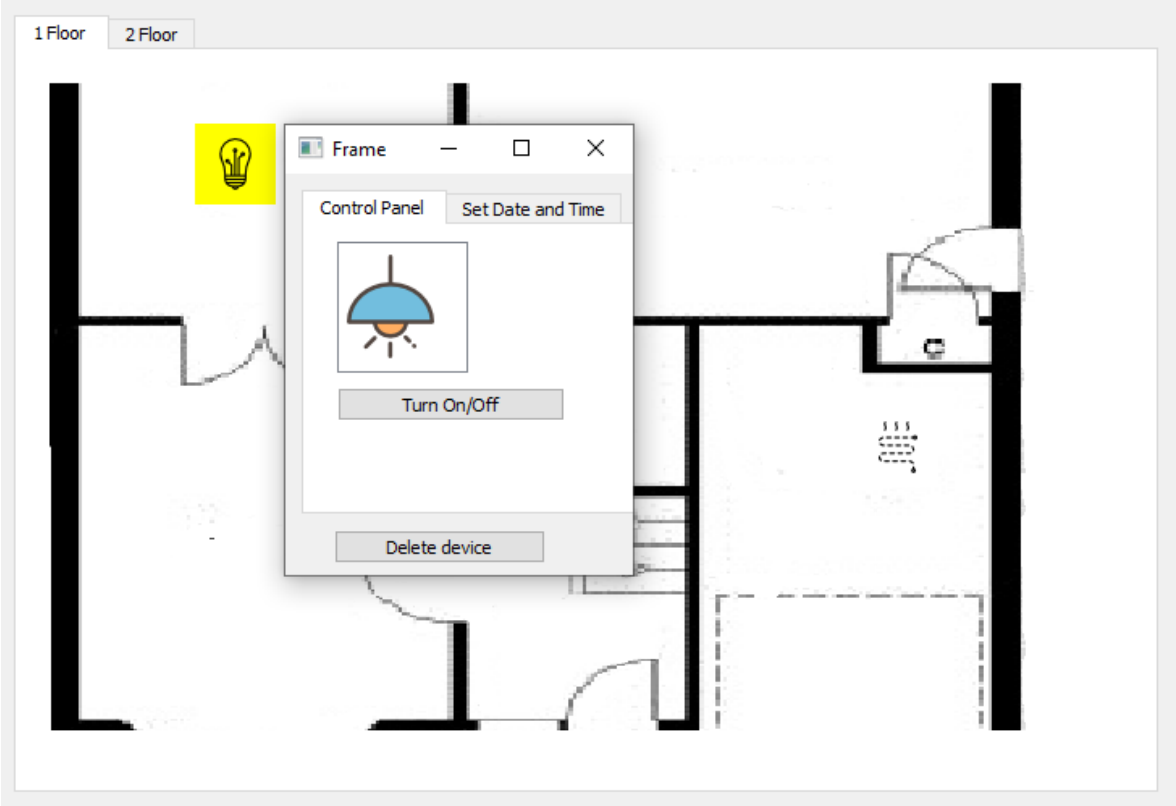


Рисунок 5.6 — Меню візуального компонента

Встановлення режиму роботи пристрою відбувається за допомогою панелі керування візуальним компонентом. Панель керування буває двох типів: для складного компонента та для простого компонента. Перший тип представляє собою слайдер для задавання значення пристрою, малюнок компонента та кнопки для оновлення виставленого слайдером значення і для увімкнення або вимкнення приладу (рисунок 5.7). Другий тип панелі управління складається тільки з малюнка компонента і кнопки для увімкнення або вимкнення приладу (рисунок 5.6).

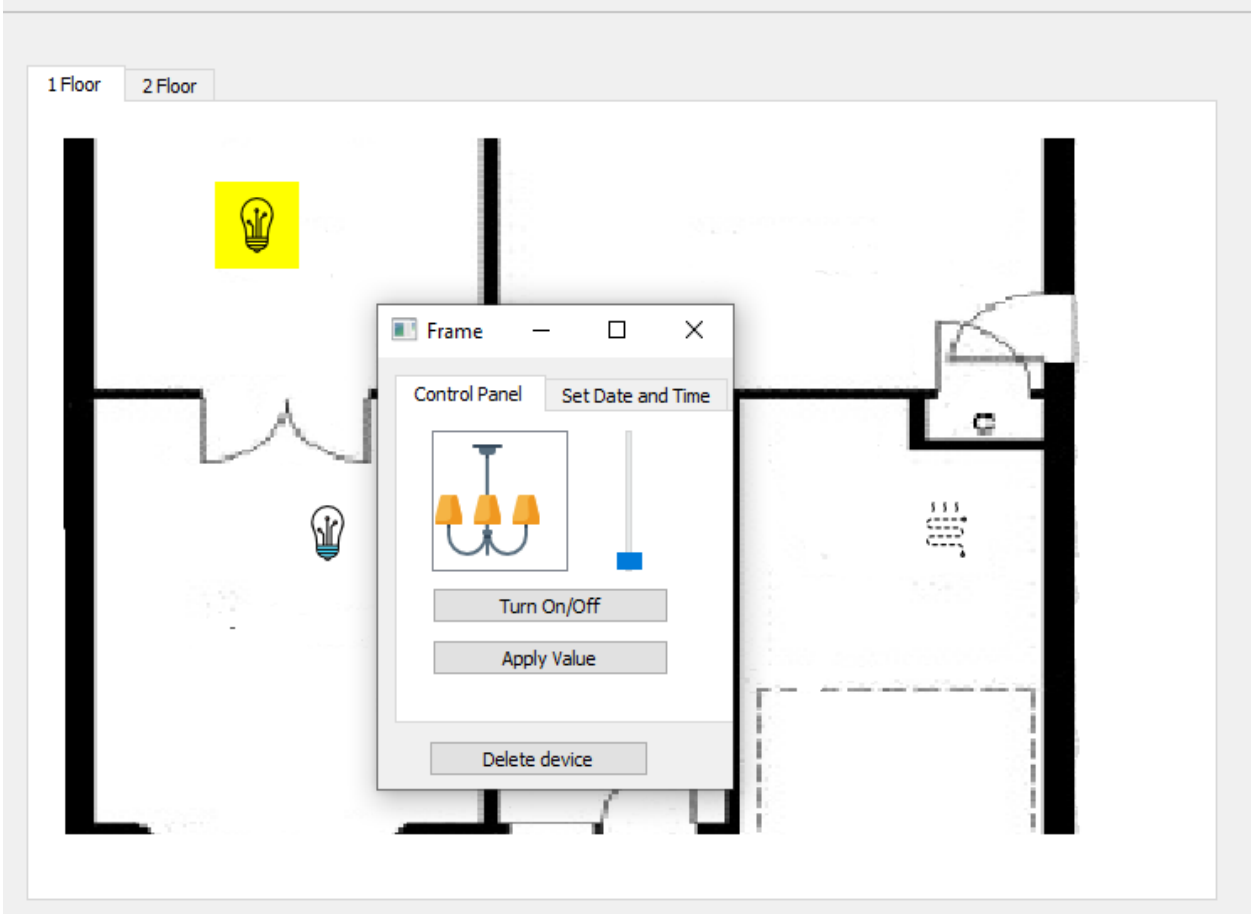


Рисунок 5.7 — Меню управління складним візуальним компонентом

Зміна режиму роботи пристрою відбувається за допомогою сигналів і слотів. Натиснувши на кнопку, спрацьовує відповідний слот, де формується сповіщення про зміну режиму роботи пристрою і відправляється серверу. Після того як сервер успішно обробив повідомлення і прислав клієнту зворотнє повідомлення, відбувається оновлюється json файлу і. змінюється візуальне представлення компонента.

Для встановлення дати та часу зміни роботи пристрою, користувачеві слід відкрити відповідну панель. Дана панель складається з поля для задавання дати, календаря для встановлення дати, слайдера для встановлення значення і кнопки для підтвердження операції (рисунок 5.8).

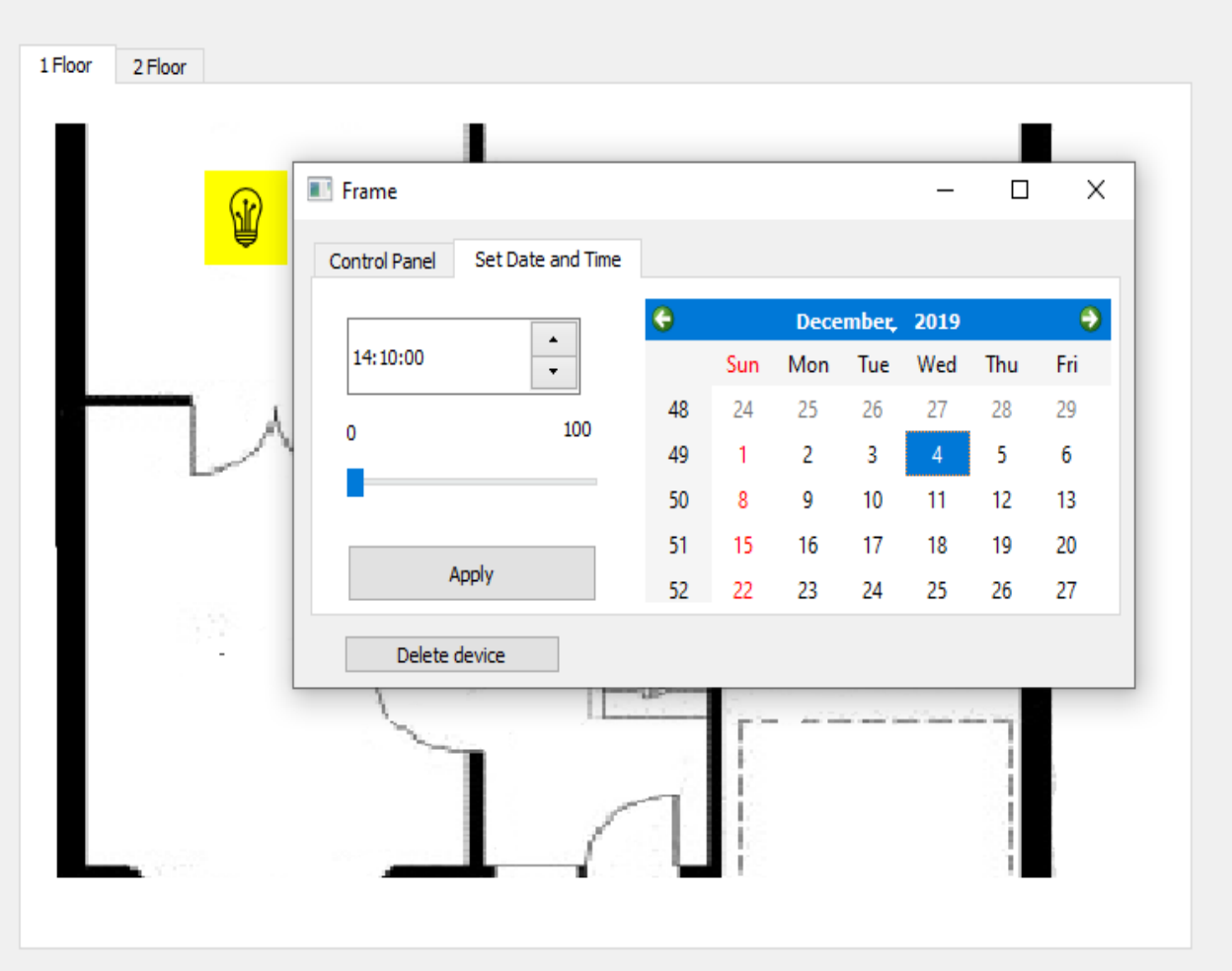


Рисунок 5.8 — Панель встановлення дати та часу зміни режиму роботи пристрою

Користувач може переміщувати компоненти, здійснивши подвійне натискання лівої кнопки миші по компоненту і затиснувши ліву кнопку миші. Для виходу з режиму переміщення користувачеві знову треба здійснити подвійне натискання лівої кнопки миші.

6. РЕАЛІЗАЦІЯ СТАРТАП ПРОЕКТУ

В даному розділі проводиться маркетинговий аналіз стартап проекту для визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

6.1 Опис ідеї проекту

У даному розділі описано економічне обґрунтування реалізації стартап-проекту на тему “Інструментальні засоби моделювання систем автоматизації споруд”. В таблиці 6.1 описано ідеї стартап-проекту.

Таблиця 6.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Сегменти споживачів
Ідея полягає у тому, щоб розробити набір інструментів для проведення певних експериментів над системами автоматизації споруд.	1. Моделювання системи розумного будинку.	Компанії, які займаються розробкою побутових приладів.
	2. Перевірка ефективності роботи приладу в приміщенні.	Компанії, які займаються розробкою побутових приладів. Господарі офісів чи будинків.

Аналіз потенційних техніко-економічних переваг ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

— визначення переліку техніко-економічних властивостей та

характеристик ідеї;

— визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

— проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні) (таблиця 6.2).

Таблиця 6.2. — Визначення сильних, слабких та нейтральних характеристик

		(потенційні) товари/концепції конкурентів			
No п/п		**Мій проект**	**Smart Things**	**Home Kit**	**Nest**
1.	**W (слабка сторона)**	Працює з обмеженою кількістю пристроїв	Містить проблеми з підключе-нням до пристроїв	Дуже висока ціна на продукцію компанії	Не підтримується в Україні. Містить проблеми з авторизацією
2.	**N (нейтра-льна сторона)**	Відсутність можливості зберігати історі про пристрої	Працює із пристроями, які сумісні із Sumsung і SmartSings	Працює лише пристроями, сумісними з продуктами Apple	Працює із пристроями, які сумісні із Nest і Google

Продовження таблиці 6.2

3.	**S (сильна сторона)**	Зберігає дані на сервері	Працює з обширним списком пристроїв. Зберігає дані на сервері	Містить зручний інтерфейс. Зберігає дані на сервері	Зберігає дані на сервері. Має зручний інтерфейс

6.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових (таблиця 6.3):

— за якою технологією буде виготовлено товар згідно ідеї проекту;

— чи існують такі технології, чи їх потрібно розробити/доробити;

— чи доступні такі технології авторам проекту.

Таблиця 6.3. —Технологічна здійсненність ідеї проекту

No п/п	**Ідея проекту**	**Технології і реалізації**	**Наявність технологій**	**Доступність технологій**
1	Створення інструментів	C++, Qt	Наявна	Безкоштовна, доступна
2	Створення сервісу	Python	Наявна	Безкоштовна, доступна
3	Розширювана база даних	MongoDB	Наявна	Безкоштовна, доступна

Продовження таблиці 6.3

Висновок: проект реалізувати можливо. Обрана технологія реалізації ідеії проекту: Створення інструментальних засобів моделювання систем автоматизації споруд.

За результатами аналізу таблиці робиться висновок щодо можливості технологічної реалізації проекту: так чи ні, а також технологічного шляху, яким це доцільно зробити (з поміж названих технологій обираються такі, що доступні авторам проекту та є наявними на ринку).

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводиться аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 6.4).

Таблиця 6.4. — Попередня характеристика потенційного ринку стартап-проекту

No п/п	**Показники стану ринку (найменування)**	**Характеристика**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн/ум.од	20000
3	Динаміка ринку (якісна оцінка)	Зростає

Продовження таблиці 6.4

4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	25%

Рентабельність — поняття, що характеризує економічну ефективність виробництва, за якої за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержується прибуток як головне джерело розширеного відтворення. Суть одного із найважливіших методів оцінки економічної ефективності інвестицій полягає у розрахунку їх середньої рентабельності за формулою

$$R = \frac{P}{1 \times n} \times 100$$

де Р - прибуток за час експлуатації проекту; / - повна сума інвестиційних витрат; п - час експлуатації проекту.

Середня норма рентабельності в галузі (або по ринку) порівнюється із банківським відсотком на вкладення. За умови, що останній є вищим, можливо, має сенс вкласти кошти в інший проект.

За результатами аналізу таблиці робиться висновок щодо того, чи є ринок привабливим для входження за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (таблиця 6.5).

Таблиця 6.5. — Характеристика потенційних клієнтів стартап-проекту

No п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Додаток для управління пристроями за допомогою візуальних компонентів	Фізичні особи, компанії, які займаються розробкою пристроїв для автоматизації будівлі	Цільова група це фізичні особи і працівники на підприємствах, відмінностей між групами не має.	стабільність роботи; невисока ціна; зручний у використанні

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (таблиці 6.6-6.7).

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку. Аналіз пропозиції необхідно виконати аналізуючи існуючі види конкуренцій.

Необхідною умовою ефективного функціонування механізму саморегулювання ринкової економіки є конкуренція. Вона є важливою рушійною силою розвитку ринкової економічної системи. Конкуренцію породжують об'єктивні умови ринкового господарювання: різні форми власності на засоби виробництва; повна економічна відокремленість і свобода вибору господарської діяльності товаровиробників, їх повна економічна залежність від кон'юнктури ринку; боротьба за джерела сировини, ринки збуту

виробленої продукції, сфери використання капіталу з метою отримання найбільшого прибутку.

Конкуренція — це суперництво (змагальність) між різними учасниками ринкової економіки за найбільш вигідні умови виробництва та реалізації товарів і послуг, за привласнення найбільшого прибутку. Вона виступає силою, яка мобілізує особистий економічний інтерес і підприємницький потенціал та спрямована на їх максимальну реалізацію.

У Законі України "Про обмеження монополізму та недопущення недобросовісної конкуренції у підприємницькій діяльності" зазначається: конкуренція - це "змагальність підприємців, коли їх самостійні дії обмежують можливості кожного з них впливати на загальні умови реалізації товарів на ринку і стимулюють виробництво тих товарів, яких потребує споживач". Боротьба на ринку великої кількості різних товаровиробників і постачальників ресурсів за споживача (покупця) і економічний успіх - об'єктивний економічний закон. Захист конкуренції, суб'єктів господарювання і споживачів від недобросовісної конкуренції передбачає демонополізацію вітчизняної економіки і створення ринкового конкурентного середовища.

Таблиця 6.6. — Фактори загроз

No п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1	Конкуренція	Вихід на ринок нових компаній.	— вихід з ринку; — об'єднатися з іншою компанією; — розширення функціоналу власног продукту.

Продовження таблиці 6.6

2	Зміна потреб користувача	Користувачам необхідні нові функції в додатку	Розширювати функціонал власного додатку.

Отже, було проаналізовано фактори загроз ринкового впровадження проекту, серед яких: конкуренція, уповільнення росту ринку, зміна потреб користувачів, зміна тарифів провайдера хмарного розгортання на платні та надходження на ринок альтернативних продуктів. Було також запропоновано можливі реакції компанії.

Таблиця 6.7. — Фактори можливостей

No п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	Зростання можливостей потенційних покупців	Зростання фінансування у підприємств, які займаються предметами побуту	Запропонувати їм свої послуги
2	Недоліки в існуючих альтернативах	Існуючі альтернативи або працюють повільно, або не орієнтованими на конкретн предметну область	Модифікація існуючих платформ

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (таблиця 6.8). Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці.

Таблиця 6.8. — Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1. Вказати тип конкуренції: чиста	Існують три фірми конкуренти на ринку	Прямі договори з стартапами, презентація продукту на виставках
2. За рівнем конкурентної боротьби: міжнародний	Компанії знаходяться за межами України	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою: внутрішньогалузева	Конкуренти мають ПЗ, яке використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях
4. Конкуренція за видами товарів: товарно-видова	Види товарів є однаковими, а саме – програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг: нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки продукції.
6. За інтенсивністю: не марочна	Бренди відсутні	-

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі (таблиця 6.9).

На основі аналізу конкуренції, а також із урахуванням характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за таблицею 6.10.

Таблиця 6.9. — Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти галузі**	**Потенційні конкуренти**	**Постачальник**	**Клієнти**	**Товари-замінники**
		Системе формування сценаріїв		Контроль якості	Лояльність споживачів
Висновки:	Визначити інтенсивність конкурентної боротьби з боку прямих конкурентів	Є можливості виходу на ринок, оскільки існуючі рішення не надають повністю всіх необхідних можливостей	Постачальники відсутні	Важливим для користу-вача є швидкість роботи ПЗ.	Товари-замінники можуть використат и більш дешеву техно-логію створення ПЗ та зменшити собівартість товару

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку.

Таблиця 6.10. — Обґрунтування факторів конкурентоспроможності

No п/п	**Фактор конкурентоспроможнос**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентни проектів значущим)**
1	Виконання ПЗ у вигляді програми у вигляді додатку	Дозволяє наочно побачити роботу ПЗ і правильність роботи.
2	Простота інтерфейсу користувача	Користувач має лише завантажити дані і запустити програму на виконання

За визначеними факторами конкурентоспроможності (таблиця 6.10) проводиться аналіз сильних та слабких сторін стартап-проекту (таблиця 6.11).

Таблиця 6.11. — Порівняльний аналіз сильних та слабких сторін

No п/п	**Фактор конкурентоспроможнос**	**Бали 1-20**	**Реийтинг товарів-конкурентів у порівнянні з Database Generator (даним продуктом)**						
			-3	**-2**	**-1**	**0**	**1**	**2**	**3**
1	Виконання ПЗ у вигляді програми у вигляді додатку	15			+				

Продовження таблиці 6.11

2	Простота інтерфейсу користувача	20							+

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиця 6.12) [22].

Таблиця 6.12. — SWOT-аналіз стартап-проекту

Сильні сторони:	**Слабкі сторони:**
простий інтерфейс користувача, виконання ПЗ у вигляді програми	необхідно мати широкий набір пристроїв, з якими додаток міг взаємодіяти.
Можливості:	**Загрози:**
додаткове держфінансування для досліджень у підприємствах, які є потенційними покупцями	конкуренція, зміна потреб користувачів

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 6.13. — Альтернативи ринкового впровадження стартап-проекту

No п/п	**Альтернатива (орієнтовний комплек заходів) ринкової поведінки**	**Ймовірність отриманн ресурсів**	**Строки реалізації**
1	Створення додатка на основі існуючих САПР	25 %	12 місяців
2	Переорієнтація на веб-розробку	40 %	6 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Томуобираємо альтернативу 2.

6.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (таблиця 6.14).

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку.

Таблиця 6.14. — Вибір цільових груп потенційних споживачів

No п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота вход у сегмент**
1	Фізичні особи	Готові	Середній	Висока	Складно
2	Підприємства	Готові	Високий	Висока	Складно
3	Державні установи	Потребують недовгих переговорів	Середній	Висока	Складно
Які цільові групи обрано: фізичні особи, підприємства.					

Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку (таблиця 6.15).

Розроблення ринкової стратегії впершу чергу передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 6.15. — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	**Стратегія охоплення ринку**	**Ключові конкурентос-проможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку***
Надання товару важливих з точки зору споживача відмітних властивостей	Стратегія диференціації	Стартапери потребують швидкості розробки, яку надає підтримка декількох платформ даним продуктом	Стратегія спеціалізації (спирається на диференціацію)

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту.

Наступним кроком є вибір стратегії конкурентної поведінки (таблиця 6.16).

Таблиця 6.16. — Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	**Чи буде компанія шукати нових споживачів**	**Чи буде компанія копіювати основні характеристики конкурента**	**Стратегія конкурентної поведінки**
Ні	Шукати нових споживачів, забирати існуючих у конкурентів	Пов'язувати та розширювати, створюючи новий функціонал	Стратегія заняття конкурентної ніші

6.5 Аналіз ринкових можливостей запуску стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач.

Таблиця 6.17. — Визначення ключових переваг концепції потенційного товару

Потреба	**Вигода, яку пропонує товар**	**Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)**
Швидкодія	Має низьку ціну	Краща стабільність роботи.

Продовження таблиці 6.17

Простота користувацького інтерфейсу.	Простота роботи з ПЗ	Користувачі мають зручний інтерфейс для взаємодії з ПЗ

Надалі розробляється трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання (таблиця 6.18).

Таблиця 6.18. — Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">ПЗ забезпечує користувача інструментами для моделювання
систем автоматизації споруд.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Швидкість
2. Ціна
3. Зручний інтерфейс</td><td></td><td>Технологічна</td></tr>
<tr><td colspan="3">Якість: тестування на предмет багів</td></tr>
<tr><td colspan="3">Маркування відсутнє</td></tr>
<tr><td colspan="3">Марка: SmartSoft</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">1-місячна пробна безкоштовна версія</td></tr>
<tr><td colspan="3">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: патент</td></tr>
</table>

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. Захист може бути організовано за рахунок захисту ідеї товару (захист інтелектуальної власності), або ноу-хау, чи комплексне поєднання властивостей і характеристик, закладене на другому та третьому рівнях товару.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів. Аналіз проводиться експертним методом.

Таблиця 6.19. — Визначення меж встановлення ціни

No п/п	**Рівень цін на товари-замінники**	**Рівень цін на товари-аналоги**	**Рівень доходів цільової групи споживачів**	**Верхня та нижня межі встановлення ціни на товар/послугу**
1	1500…4000 грн	1800…6000 грн	25000…50000 грн	200…500 грн

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (таблиця 6.20).

Таблиця 6.20. — Формування системи збуту

No п/п	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна систем збуту**
1	Купують підписку та роблять щорічні внески для подовження ліценції	Продаж	Однорівневий	Власна та через посередників

Останньою складової маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (таблиця 6.21)

Таблиця 6.21. — Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	**Канали комунікації, якими користую-ться цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлен-ня**	**Концепція рекламного звернення**
Купівля ПЗ через інтернет, робота з ПЗ на комп‘ютера х з різними ОС	Інтернет	Швидкодія, простота використання	Показати переваги ПЗ, у тому числі і перед ко-нкурентами	Демо-ролик із викори-стання

Розроблений програмний продукт має переваги над існуючими конкурентами та є конкурентноздатним на ринку. Програма має шляхи подальшого розвитку, визначені маркетингові стратегії та шляхи збуту. Основна цільова аудиторія – підприємства, державні установи та домовласники.

ВИСНОВКИ

Програма представляє собою набір інструментальних засобів для моделювання систем автоматизації споруд. Для вирішення даної задачі були виконані наступні задачі:

— розроблено бібліотеку візуальних компонентів. Бібліотека візуальних компонентів представляє собою набір графічних компонентів (віджетів) із функціоналом для керування пристроїв;

— створено засоби для розробки сценарію керування пристроями автоматизованої споруди;

— реалізовано збереження даних моделі та синхронізацію з пристроями;

— виконано тестове моделювання системи автоматизації споруди.

Даний продукт складається з серверної та клієнтської частин, модулю керування пристроями та сервера бази даних.

Клієнтська частина написана на мові C++ на основі фреймворку Qt. Завдяки такому підходу, вдалося досягти поставлених цілей, таких як: автоматизація пристроїв в будинку, можливість розширення бібліотеки новими компонентами, можливість використання в різних системах та на різних платформах. Цьому також сприяє використання шаблонів проектування таких як Abstract Method, Prototype і MVC та технології об’єктно-орієнтованого програмуваня.

Серверна частина реалізована на мові програмування Python. Даний вибір інструментів дозволив легко реалізувати синхронізацію даних моделей з сервером бази даних MongoDb. Взаємодія між сервером та користувачем реалізована через протокол TCP/IP.

Модуль керування пристроїв відповідає за управління пристроями і реалізований лише у вигляді інтерфейсів.

Графічний інтерфейс даної системи розроблено у вигляді додатку за допомогою Qt Designer. Це сприяло покращенню навичок володіння функціоналом Qt Designer. Для розробки інтерфейсу використовувались віджети різного типу.

Дана програма призначена для домовласників, підприємств, які займаються розробкою побутових пристрої для будинків, а також для науковців для поставлення певниї експериментів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лафоре Р. Обьектно-ориентированное программирование в C++ / Р. Лафоре. — Москва: Питер, 2004. — 922 с.
2. Stroustrup B. The C++ Programming Language / Bjarne Stroustrup. Boston: Addison-Wesley, 1991. — 369 с.
3. Prata S. C++ Primer Plus / Stephen Prata. — Boston: Addison–Wesley, 2012. — 1438 с.
4. Design Patterns, Elements of Reusable Object-Oriented Software / E.Gamma, R. Helm, R. Johnson, J. Vlissides. — Boston: Addison–Wesley, 1994. — 395 с.
5. Джонс А. Программирование в сетях Microsoft Windows / А. Джонс, Дж. Оланд. — Москва: Питер, 2002. — 601 с.
6. Бланшет Ж. QT 4: программирование GUI на C++ / Ж. Бланшет, М. Саммерфилд. — Санкт-Петербург: Кудиц-Пресс, 2007 — 628 с.
7. Hows D. MongoDB Basics / David Hows, Peter Membrey, Eelco Plugge. — New York: Apress, 2014. — 117 с.
8. Стивенс Р. Алгоритмы. Теория и практическое применение / Род Стивенс. — Москва: Эксмо, 2017. — 544 с.
9. Скиена С. Алгоритмы. Руководство по разработке / Стивен Скиена. — Санкт-Петербург: БХВ-Петербург, 2017. — 720 с.
10. Уорен Г. Алгоритмические трюки для программистов / Генри Уорен. — Киев: Вильямс, 2014. — 504 с.
11. Седжвик Р. Фундаментальные алгоритмы на C++. / Роберт Седжвик. — Киев: Диасофт, 2002. — 688 с.
12. Страуструп Б. Язык программирования C++. Стандарт C++11. Краткий курс / Бьерн Страуструп. — Москва: Бином, 2017. — 176 с.

13. Джонс Э. Программирование в сетях Microsoft Windows / Э. Джонс, Д. Оланд. — Москва: Питер, 2002. — 594 с.
14. Шлее М. Qt5.10. Профессиональное программирование на C++ / Макс Шлее. Санкт-Петербург: БХВ-Петербург, 2018. — 1072 с.
15. Форум програмістів [Електронний ресурс] — Режим доступу до ресурсу: https://stackoverflow.com/.
16. Довідник по шаблонам проектування [Електронний ресурс] — Режим доступу до ресурсу: http://cpp-reference.ru/patterns/.
17. Довідник по C++ [Електронний ресурс] — Режим доступу до ресурсу: http://www.cplusplus.com/reference/.
18. C++ документація [Електронний ресурс] — Режим доступу до ресурсу: http://devdocs.io/cpp/.
19. Мережа розробників Microdoft [Електронний ресурс] — Режим доступу до ресурсу: https://msdn.microsoft.com/ru-ru/library.
20. Qt документація [Електронний ресурс] — Режим доступу до ресурсу: http://doc.qt.io/.
21. Мартин Ф., UML. Основы/ Скотт Кендал, Мартин Фаулер. — Санкт-Петербург: БХВ-Петербург, 2005 — 167 с.
22. Харниш, В. Правила прибыльных стартапов: как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва: Манн, Иванов и Фербер, 2012. – 279 с.
23. Бариленко В. И. Бизнес-анализ как важный вид консалтинговых услуг // РИСК: Ресурсы, Информация, Снабжение, Конкуренция.— №4.— 2012.— С.202-207.
24. Калянов Г.Н. Теория и практика реорганизации бизнес-процессов. — М.: СИНТЕГ, 2000. 212 с.
25. Квашнин А. Как продвигать проекты коммерциализации технологий: серия методических материалов «Практические руководства для

центров коммерциализации технологий» / М. Катешова, А. Квашнин, под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 52 с.

26. MongoDB документація [Електронний ресурс] — Режим доступу до ресурсу: https://docs.mongodb.com/manual/reference/.
27. MongoDB документація [Електронний ресурс] — Режим доступу до ресурсу: https://www.tutorialspoint.com/mongodb/index.htm.
28. Michael J. TCP.IP Sockets in C / Michael J.Donahoo, Kenneth L.Calvert — Boston: Addison–Wesley, 2003. — 365c.
29. Messerschmitt. D. Networked Applications: A Guide to the New Computing Infrastructure/ David G. Messerschmitt — Boston: Addison–Wesley, 1999. — 280c.
30. Giamas A. Mastering MongoDB 4.x-Second Edition/ Alex Giamas — New York: Apress, 2019. — 394c.